Universidad
Carlos III de Madrid

ANÁLISIS Y OBTENCIÓN DE GRANDES VOLÚMENES DE DATOS EN REDES SOCIALES

Trabajo de Fin de Grado.

Autor: Javier Serrano Valleros

Director: Ignacio Martín Martínez

Codirector: José Alberto Hernández Gutiérrez

Grado en ingeniería telemática

Septiembre 2016

ABSTRACT:

El objetivo de este proyecto es describir y reportar los logros en la investigación y desarrollo en el campo de Data Analytics. El documento describe el proceso de investigación y procesado de la información de una red social del campo de la restauración, conocida como “El Tenedor”.

La memoria cubre el proceso de la selección y obtención de datos a través de tecnologías emergentes tales como Python o MongoDB, el procesado de Datos utilizando el framework de R que cuenta con algoritmos como TF-IDF o Estimadores Lineales. Finalizando con una representación de los datos en entornos como CartoDB.

La mayor parte de los resultados son extraídos a través de simple métodos estadísticos aplicados a los datos relevantes encontrados en el Tenedor, tales como los comentarios, las notas de los usuarios hacia el restaurante, los precios y localizaciones. Además, el proyecto está diseñado para permitir el desarrollo de futuros proyectos, bien sea con datos ya obtenidos y no usados en el análisis o con información adicional.

Contenido

1 Introducción y motivación (Eng)

1.1 Motivation

We live in the information society, which offers a lot of advantages for knowledge development. In fact, there is an excess of information on internet, and an example of this is any forum on internet. You can always see several pages with plenty of comments, and a really low percentage of users reads them all. It is impossible, that is the reason why new generation tools are required and enterprises are already changing their minds to fulfill this need.

There are examples of new generation business that have already noticed of this, such as Google, that has been the first enterprise in having demonstrated profit, moreover is proved that the massive interest in Big Data by enterprises started when google announced the improvement of his economic benefits thanks to this technology. Its top application right now is Google ads, which provides personalized advertisements from a personal and very detailed profile of the user, they obtain it by comparing similar profiles and taking into account plenty of parameters from your Google searches and account, the enterprise published that they process around 20 petabytes per day. This requires a lot of personal coalification and powerful software and devices.

According to IBM [1], each day 2.5 quintillion of bytes are generated on internet, that is an opportunity for learning, the techniques that today are at the top, tomorrow are outdated and investing time and money on Big Data research can be translated on evolution to the next generation on the first place and obviously the monetary benefits that comes with it.

The aim of this project is to combine three of the trending topics nowadays, which are social networks, Big Data, and eCommerce or an approach to a real business orientation. Imagine an application that can read the whole thread of the restaurants forum, and could give as feedback the comment of comments or at least a summary of all the information that would take you a lot of time to read and taking into account parameters that you may not have noticed, so you can choose the best restaurant or at least the one where your expectations are more likely to be fulfilled, for example a restaurant where the people is more pleased, the food is better respect to the price, etc... This would summarize gigabytes of information in several interesting Megabytes, which implies not just less time for the user, also less requests for the server and a significant decrease of the network load.

To achieve our objective, we have to consider the Spanish law about data protection LOPD [2], since we could obtain user information that is related to a person, and the identity cannot be revealed, but obtaining personal data is not the objective.

1.2 Objetives

Taking into account the motivation previously exposed, the aim of this work is to obtain, study, analyze and represent the information of a social network, El Tenedor for that purpose we will follow this steps:

1. The first step is to obtain the data, in order to have a great span of alternatives, we need to get as much useful information as we can from the web application and store it in a data base.

2. Secondly we will proceed to filter and statistically analyze the data using Big Data technologies obtained and connect the data processing tool with the data base, so we can manipulate our data

3. The third step is the study and implementation within the framework information processing and retrieval algorithms to analyze the dataset and extract concrete results, conclusions and models from this analysis.

4. The last step is to represent the data analyzed in the project using several techniques such as geolocalization, wordclouds or boxplots.

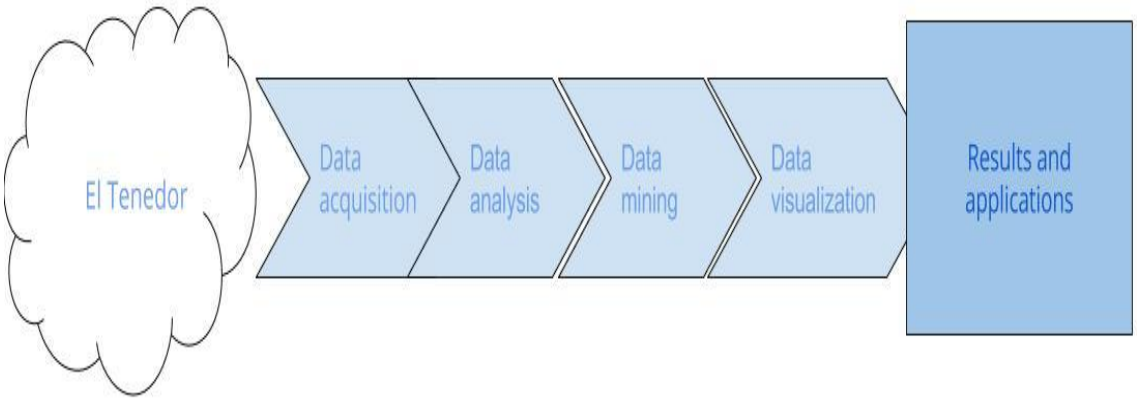


Ilustración 1 Application process

2 Estado del Arte

2.1 Técnicas de extracción de datos

La extracción puede ser llevada a cabo con diversas herramientas tales como aplicaciones web, ejemplo de ello son la extensión web Scraper de Google Chrome o scripts programados.

2.1.1 Crawler:

Un Crawler es un programa de software que navega a través de la web buscando la información deseada de manera recursiva, a través de los hiperenlaces de cada página. Su funcionalidad básica es la descarga del contenido HTML para un posterior proceso de parseo, basado en la búsqueda de tags de XML a partir dee en rutas de XPath.

Para llevar a cabo la fase de crawling, existen infinidad de herramientas. En la actualidad, entre las más populares se encuentran librerías de JavaScript, Java, R o Python.

Java posee herramientas avanzadas, que ofrecen la posibilidad de obtención de contenido HTML y parseo simultaneo, ejemplo de esto son crawler4j, webSPHINX o JSpider. Estas herramientas permiten al programador obtener toda la información de una web concreta de manera recursiva. Algunas de ellas son “open source” y ofrecen soporte por parte de la comunidad de programadores.

En JavaScript existen librerías basadas en el protocolo HTTP, que realizan peticiones al servidor de la página, con métodos tales como XMLHttpRequest. Lo que permite descargar el contenido de la web, procesado con la API de HTML DOM, ésta última API es bastante compleja, añadiendo trabajo al programador, por lo que es aconsejable tener más opciones en cuenta.

R posee librerías con bastante potencial para la obtención de datos, como lo son XML o RCurl, con las que, podemos mandar una petición HTTP para descargar el contenido HTML parsear la información del HTML.

Python se está volviendo más popular entre el colectivo de expertos en Data Science para la tarea de obtención de datos, por su alta variedad de librerías. Algunos ejemplos de las librerías útiles para crawling son urlib, que abre una URL específica y descarga su contenido HTML, *HTMLParse*r o “*re”*, que parsean el documento descargado en busca de coincidencias en las expresiones regulares especificadas, o módulos más avanzados que permiten un parseo más rápido como Beautiful Soup que aplica un parser lxml.

2.1.2 Rest API

Hacer crawling vía fuerza bruta a gran escala resulta ineficiente, ya que a la hora de realizar parseo de una página HTML. Él árbol puede llegar a ser muy complejo aumentando el tiempo de operación. Debido a este problema existen ciertos servicios web, que ofrecen una herramienta para la adquisición de sus datos, denominada API REST.

Una API REST (Representational State Transfer Application Programming Interface) es una aplicación web que recibe peticiones HTTP a rutas específicas en su dominio y respuestas con información estructurada. Dicha información puede ser devuelta en varios formatos, como XML o urlencoded, aunque en la actualidad el formato JSON [3] (JavaScrip Object Notation) está ganando más relevancia. La información devuelta puede ser paginada para mejorar la eficiencia.

El modelo REST se está volviendo aún más popular debido a que ofrece información fácil de manejar a cualquier petición y es soportado en cualquier plataforma, por lo que es una buena alternativa para los servicios web que soportan páginas web y aplicación móvil.

Con intención de mejorar la seguridad de los servicios ofrecidos, varias APIs buscan protección, una solución es OAuth. Oauth es un método de autorización de tres vías desarrollado para proporcionar a un usuario mecanismos para controlar las aplicaciones que acceden a sus datos y conceder o revocar dichas autorizaciones dentro de una aplicación.

En la actualidad, varias compañías de internet ofrecen herramientas como ésta a todos los públicos. Normalmente la información ofrecida está limitada tanto por la propia autorización del usuario y por la empresa, utilizando medidas como límites de consultas diarias o registro de las aplicaciones que utilizan dichas herramientas. Entre las empresas populares que ofrece estos servicios se encuentran algunas como Twitter, Facebook o LinkedIn.

2.1.3 Herramientas web

Existen más opciones para la labor de crawling, en la actualidad se han desarrollado servicios web o aplicaciones muy avanzadas que nos permiten realizar este trabajo sin necesidad de tener conocimientos en la materia, aunque, por otro lado, cabe destacar que no son herramientas open source. Ejemplo de esto son la extensión Scraper de Chrome o una de las aplicaciones más populares en la actualidad Screaming Frog (SEO Spider Tool).

Scraper [4] permite obtener los fragmentos deseados del código HTML o el path del árbol HTML de la página sin necesidad de descargar todo su contenido además es útil en momentos determinados ya que guarda la búsqueda y los parámetros de selección para realizar búsquedas posteriores optimizadas. El problema principal es que no es escalable, aunque para pequeñas cantidades de datos ofrece comodidad de uso.

Screaming Frog [5] es una aplicación profesional, utilizada por empresas como Google, Apple o Amazon. El potencial de ésta aplicación es inmenso ya que analiza las respuestas de las peticiones HTTP para encontrar links caídos, posee redirección automática a las URL encontradas en el documento con las características especificadas, elimina el contenido duplicado y descarga el contenido útil. Además de estar conectado con Google analytics, lo que incrementa su potencial en el sector comercial. Por otro lado, se trata de una aplicación de pago y aunque sus características básicas sean gratuitas no son suficientes para un uso intensivo o con fines lucrativos, además el número de URLs accesibles con la versión gratuita están limitados.

2.2 Sistemas de almacenamiento

El almacenamiento de datos es clave para el análisis de grandes cantidades de datos, por lo que se pretende minimizar el tiempo de cada consulta en la base de datos, ya que éste es uno de los cuellos de botella más comunes, entre los procesos de extracción y análisis.

2.2.1 MongoDB

Es una de las bases de datos no relacionales más populares disponibles se refiere. Está orientada a documentos, con formato JSON, lo cual hace compatible a MongoDB con muchos lenguajes de programación. Mongo ofrece replicación de datos, alta disponibilidad y permite el escalado horizontal de manera transparente, para que la inserción de datos sea óptima.

MongoDB es soportado por Node.js, R, Python y Java, entre otros, mediante librerías conectoras como RMongo, que permiten utilizarlo a modo de cliente. El contenido de la base de datos está asegurado y es portable, ya que Mongo está provisto de un buen sistema de back up y de restauración de información.

Gracias a la replicación horizontal de mongo y el soporte, las consultas a la base de datos de mongo son muy rápidas.

En resumen, Mongo es un sistema open source de bases de datos altamente eficiente para grandes cantidades de datos, tiene una gran comunidad detrás que la respalda, lo que conduce a una de las mejores opciones para trabajar en proyectos de Big Data.

2.2.2 Cassandra

El proyecto de Apache Casandra es una NoSQL Base de datos, similar a mongoDB que ha demostrado ser altamente eficiente en sistemas distribuidos. Los datos en Cassandra son guardados conforme a un clave/valor, y la comunicación entre nodos está basada en P2P, lo que da lugar a una alta redundancia.

2.2.3 SQL

SQL es un lenguaje de gestión de bases de datos relacionales, que son aquellas cuyas tablas están relacionadas entre sí, ya sea por una o más columnas, permitiendo estructuras complejas de datos.

La base de SQL son consultas a un servidor a través de un manager. Estos managers se encargan de interpretar la sentencia de SQL, y realizar las operaciones pertinentes para el almacenamiento o visualización de datos. Entre los DBManagers más populares se encuentran MySQL, PostgreSQL, Oracle o SQL Lite, entre otros.

- Las bases de datos relacionales se guian por los siguientes puntos:
- Atomicidad: Las operaciones se ejecutan en paquetes indivisibles, que pueden realizarse con éxito o no.
- Consistencia: Una vez guardado, los datos han de ser consistentes en todas las consultas y actualizaciones.
- Aislamiento: Las consultas sobre la misma tabla han de hacerse debidamente y sin errores, es decir cada consulta sobre dicho dato ha de ser independiente, para ello existen los cerrojos.
- Durabilidad: Todos los cambios han de ser guardados satisfactoriamente.

La ventaja principal de SQL es su robustez y usabilidad.

2.3 Herramientas para la minería de datos

Un punto clave en el análisis de datos es el procesado de la información. Habitualmente los datos son guardados en la memoria del sistema de almacenamiento, y la información es procesada secuencialmente por un programa. Debido a estos parámetros se requiere un hardware potente o una gran cantidad de tiempo para obtener una aplicación eficiente.

Teniendo en cuenta los problemas que plantea la programación tradicional, se ha de optar por una estrategia más avanzada, en el que la paralelización de procesos esté muy cuidada, esto es, algoritmos que optimicen el uso de los múltiples núcleos de procesado, dividiendo las tareas en fragmentos más simples e independientes para que cada núcleo pueda trabajar de manera independiente.

Uno de los paradigmas más importantes para el procesado de datos es Map-Reduce. Éste algoritmo está basado en programación en paralelo, y lo que permite al usuario distinguir entre dos etapas, map y reduce, de esta forma se procesa la información que recibe pares de valores clave y emite los nuevos. Esta forma de programar hace la ejecución de tareas muy rápida, aunque suele añadir complejidad a los programas.

2.3.1 R

El lenguaje de programación R es uno de los programas open source más utilizados para el análisis estadístico de la información. R tiene una gran variedad de funciones y extensión de éstas con librerías programadas de manera nativa en C, lo que aumenta su eficiencia.

Partiendo de que la información ya está obtenida y guardada, R nos ofrece la posibilidad de cargarla en una estructura denominada DataFrame. Un DataFrame es básicamente una matriz con filas y columnas nombradas, pero cuyo contenido puede estar compuesto de muchos tipos diferentes de datos, lo que ofrece mucha versatilidad a la hora de utilizarlo. R está provisto de métodos que son capaces de manejar los DataFrames con eficiencia, y la mayoría de paquetes de R los acepta como una entrada.

Como conclusión podemos afirmar que R es un candidato muy a tener en cuenta para la minería y procesado de datos. Especialmente teniendo en cuenta el alcance de éste proyecto. Las librerías que tomarán más relevancia serán*: tm*, *RMongo*, *RHadoop* y *RHype*, la primera es una librería de minería de datos y procesado, y las demás son conectores con R.

2.3.1.1 tm

El paquete *tm* es básicamente una librería creada para la minería de texto que permite al usuario preparar y procesar textos fácilmente. Está basado en una creación

de Corpora, que es una colección de documentos que *tm* une, y procesa de acuerdo con las necesidades del usuario y lo optimiza para la creación de matrices de frecuencia. Además, incluye operaciones como limpieza de texto, eliminación de "palabras vacías", es decir, palabras que no aportan valor semántico a un texto tales como los nexos, basada en listas predefinidas de muchos lenguajes.

2.3.1.2 Wordcloud

El paquete Wordcloud permite relizar nubes de palabras de una manera sencilla, a partir de un conjunto de palabras en función de una frecuencia. A la hora de aplicar esto a un texto es recomendable utilizar la librería tm, para el filtrado y el orden de las palabras.

2.3.2 Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y programación funcional. Usa tipado dinámico y es multiplataforma.

En los últimos años, Python ha cobrado fuerza y se ha vuelto muy popular entre los programadores, lo que le ha provisto de una gran cantidad de librerías y herramientas para funciones muy dispares. En cuanto a Big Data se refiere, existen herramientas tales como el cambio del sistema MapReduce al procesado del lenguaje de programación natural y el soporte para estas herramientas.

2.3.2.1 Natural Language Toolkit

Natural Language Toolkit (NLTK) es una plataforma para soportar la creación de programas que controlan el análisis del lenguaje natural (humano). Provee una interfaz sencilla con más de 50 fuentes léxicas, además de buenas librerías para procesado de texto, con funciones como la tokenización, clasificación o parseo.

Gracias a su facilidad de uso y documentación se ha convertido en una herramienta común en temas de ingeniería, educación e incluso por estudiosos de la lengua.

2.3.2.2 TextBlob

La librería TextBlob es una herramienta para el procesado de texto, utilizando una simple API que complementa otras herramientas como NLTK. TextBlob proporciona funciones para clasificación, parseo, sentiment análisis, tokenización y operaciones por frecuencia de palabra o frase.

2.3.2.3 Pandas

Lo que ofrece Pandas son estructuras simples de datos y herramientas para procesado de datos de alto rendimiento. Esta librería por sí sola no aporta un entorno de análisis para Python, pero combinándolo, hace de Python una herramienta muy poderosa para el análisis.

2.3.2.4 Octo

Octo proporciona una buena optimización para tareas en las que se necesita separar operaciones en grandes datases en varias tareas paralelas. El sistema está basado en un modelo cliente-servidor, donde el servidor coordina todos los clientes conectados, donde ejecutará las funciones definidas por el usuario con los datos almacenados en el servidor. Cuando cada cliente haya finalizado, todos los valores de resultado retornarán al servidor y se finalizará la ejecución. Una vez finalizada la tarea los clientes esperarán nuevas tareas del servidor.

2.3.3 Ecosistema Hadoop

Apache Hadoop es entorno de software open source que soporta aplicaciones distribuidas. Lo que ha hecho a Hadoop convertirse en una herramienta apta para el Big Data es su sistema de almacenamiento distribuido, es decir, los datos tratados por el sistema Hadoop son fragmentados y replicados, siendo capaz de coordinar donde están localizados para un uso más eficiente de los datos.

Hadoop está compuesto por dos componentes principales: Hadoop Distributed File System y MapReduce. Ambos componentes se basan en la programación distribuida y comunicación continua entre una red de nodos. Las operaciones de MapReduce suelen tomar como entrada un directorio de archivos almacenados en el sistema de archivos Hadoop y calcula las operaciones necesarias de una manera distribuida.

2.3.3.1 Hadoop Distributed File System

Hadoop Distributed File System (HDFS) es el componente de almacenamiento de la plataforma Hadoop. HDFS consiste en un conunto de nodos distribuidos que almacenan información y sus réplicas, los principales componentes de HDFS son:

- NameNode: Es el nodo primario del sistema de archivos Hadoop y está a cargo de hacer el seguimiento de todos los documentos almacenados en el sistema. Almacena los metadatos relacionados con todos los archivos y es el encargado de iniciar y coordinar las operaciones de archivo básicas tales como la apertura, creación o modificación de los clientes el NameNode proporciona al usuario un sistema consistente de manejo de

archivos y determina el mapeo de los bloques de información a los DataNodes.

- NameNode Secundario: Se encarga de la creación de puntos de comprobación del NameNode para evitar la pérdida de datos. Aunque no se puede considerar un nodo de redundancia, es decir en caso de fallo del NameNode, no se asegura que el sistema funcione correctamente, aunque guarde una copia de los metadatos para la recuperación de éste.
- Datanode: Es el encargado de manejar las peticiones de escritura y lectura al usuario. En ellos se guardan los fragmentos de la información, siendo coordinados por el NameNode. En resumen, los DataNodes no poseen lógica, solamente leen o escriben los fragmentos en función de las peticiones.

Los nodos mencionados previamente son agrupados en clusters. Cada cluster debe contener al menos un NameNode y uno o más DataNodes, dichos clusters han de ser instalados en diferentes máquinas conectadas por red o en una sola, utilizando software para simular el sistema distribuido.

2.3.3.2 Hadoop MapReduce

Hadoop proporciona una API para que las funciones de MapReduce sean tomadas como entradas de texto guardadas en el HDFS y las salidas sean guardadas en otro directorio de HDFS. El código Python para el desarrollo usa la librería de Hadoop Streaming API

Cada función de mapeo o reducción se codifica en diferentes ficheros o programas Python (mapper.py o reducer.py respectivamente). Una vez definido todo el usuario puede hacer funcionar la función de MapReduce usando la API de Hadoop Streaming, pasando como parámetros la localización de las funciones de mapeo y reducción escritas en Python.

2.3.3.3 RHadoop

Es un conector de Hadoop con R, desarrollado por Revolution analitycs, ahora Microsoft R. Hadoop es una herramienta poderosa, aunque se ve limitada debido a la imposición de necesitar programas Java para realizar la función de map-reduce.

A priori no parece una gran desventaja, al no ser un lenguaje diseñado para desempeñar labores estadísticas como R. Es por esto que existe la librería RHadoop, que permite realizar las funciones de map-reduce en un entorno más apropiado, RHadoop está compuesto por tres librerías, *rmr*, *rhdfs* y *rhive*.

El paquete *rmr* es el encargado de la conexión de la función de map-reduce de hadoop con las tareas definidas en código R, a pesar de trabajar en R, este paquete tiene el control del cluster de Hadoop y lleva a cabo todas las tareas que utilizan todos los demonios y recursos de Hadoop, no sólo los reservados para R.

Rhdfs es un paquete de interfaz entre Hadoop Distributed File System y el entorno de R. Básicamente permite al sistema de ficheros deHadoop ser contrado desde la consola de R. Este paquete ofrece una interfaz a HDFS con funciones básicas y avanzadas para los datasets almacenados en HDFS

Rhbase es una interfaz para operaciones con la fuente de datos guardada en la red distribuida Hadoop HBase. Proporciona varios métodos para la inicialización, lectura, escritura y manipulación de operaciones sobre las tablas.

Sin embargo, no es necesario instalar todos los paquetes para que funcione MapReduce en R. Utilizando HBase, necesitaríamos rhbase. Sino con instalar rmr para MapReduce y rhdfs para la interfaz de HDFS sería suficiente.

2.3.3.4 Rhipe

Es un conector entre R y Hadoop que permite las funciones de map-reduce utilizando R. El funcionamiento de Rhipe consiste en un cliente de R, es decir un programa RClient, que mapee y envíe las funciones de Hadoop al Hadoop Job Tracker, pasando todos los parámetros necesarios.

Uno de las claves en la librería de RHipe es que proporciona una interfaz sobre Hadoop. Cualquier usuario de R puede realizar operaciones sobre grandes datases almacenados en HDFS, lo que permite al usuario disfrutar del almacenamiento en Hadoop y las ventajas de R.

2.3.4 Spark

Spark es una tecnología open source para el análisis de datos construido sobre el sitema de ficheros distribuidos de Hadoop (HDFS). Spark no utiliza el clásico sistema MapReduce, llegando a mejorar su eficiencia en condiciones ideales hasta cien veces.

La clave para el rendimiento de spark es que proporciona una gran eficiencia trabajando con clusters en memoria. Además, existen APIs de Spark para Java, Python y Scala lo que lo hace más fácil de trabajar con él

Cada aplicación de Spark funciona como un conjunto de procesos independientes en un cluster, coordinados por el objeto SparkContext en el programa principal. El SparkContext puede conectarse a múltiples tipos de managers para clusters, como Hadoop YARN o propios de Spark para colocar los recursos de las aplicaciones.

2.3.5 Matlab

Matlab es un software matemático que ofrece un entorno de desarrollo integrado para el desarrollo de computaciones de matrices, funciones u otros elementos matemáticos a través de scripts en un lenguaje de programación llamado M.

Es considerado una solución viable ya que ofrece una variedad de librerías que contienen soluciones para un sinfín de problemas en el ámbito de la ingeniería, en áreas como las telecomunicaciones, transformaciones matemáticas, procesados y análisis de imagen y sonido e incluso herramientas para el análisis estadístico.

Sin embargo, el problema de Matlab comienza cuando se procesan grandes cantidades de datos, ya que el procesado no está optimizado ni las herramientas estadísticas tampoco, no presenta una solución tan buena como lo puedan ser otras al no ser escalable.

2.3.6 Scala

Scala es un lenguaje de programación multi-paradigma diseñado para expresar patrones comunes con tipos seguros. Tiene integrado características de lenguajes funcionales y orientados a objetos, la implementación actual funciona en una máquina virtual Java y es compatible con aplicaciones Java.

La ventaja principal que aporta Scala es la escalabilidad como resultado de su integración de los paradigmas de programación orientada a objetos y a funciones. Al ser orientado a objetos permite la creación de componentes más complejos en la arquitectura a través de clases mientras que al ser también funcional ofrece un soporte para dar el paso de un “java sin punto y coma” a composiciones funcional de patrones, siempre dando soporte a las preferencias del programador.

También se ha de tener en cuenta que Scala puede operar con Java. Las clases pueden estar mezcladas y con referencias cruzadas, ya que el compilador de Scala incluye un subconjunto del compilador de Java para permitir estas mezclas. Además todo en Scala es un objeto, lo que ofrece una flexibilidad necesaria para hacer que un lenguaje evolucione y sirva para crear software para servidores usando procesamiento concurrente y síncrono, procesamiento paralelo o distribuido en un entorno como la nube.

2.4 Tecnologías web

A la hora de desarrollar servicios web, existe una gran variedad de alternativas entre los lenguajes de programación y plataformas web que facilitan la tarea de su creación.

2.4.1 Node.js

Node es un entorno multiplataforma, de código abierto programado para la creación de programas de redes altamente escalables, como servidores web y es una

adaptación local de Javascript. Además, cuenta con npm que es un manejador de paquetes para Node.js lo que facilita el acceso a las herramientas web.

El entorno Express es un paquete que facilita el desarrollo web. Express se utiliza sobre todo para programación back-end. Si se utiliza junto a npm provee el código básico para un host funcional que puede ser modificado para convertirse en una aplicación web de alto nivel. Además, cuenta con Jade, que es un motor de templates HTML basada en javascript.

También se puede tener en cuenta sails.js que proporciona ORM, autogeneración de las APIs REST, funciones de seguridad básicas. Otros ejemplos que ofrecen funcionalidades similares son Rhapsody.js, Total.js o Locomotive.

2.4.2 PHP

PHP es un lenguaje interpretado de back- end orientado a objetos, muy útil en desarrollo web, ya que ofrece varios entornos para crear servicios web simples. Funciona en un servidor apache y es de código abierto.

PHP se introduce dentro de un fichero html y es interpretado en el servidor antes de enviar la respuesta al usuario. Además, PHP ofrece una documentación muy completa y posee una comunidad muy buena para los desarolladores.

2.4.3 Django

Django es un entorno para Python que lo hace aún más sencillo de utilizar ya que ofrece un conjunto de funciones predefinidas que ahorran una gran cantidad de tiempo al programador. También incluye ORM y autogenaración de las APIs REST.

Por último, cabe destacar que Django ofrece plantillas para código html y acceso a ficheros y recursos externos.

2.4.4 HTML5

La quinta versión de HyperText Markup Language es conocida como HTML5. El objetivo principal del lenguaje es ayudar a definir el estado y límites de los elementos dentro de un buscador web. La ventaja que tiene respecto a sus predecesores es que permite a los programadores implementar funciones modernas tales como la de arrastrar y soltar, validación de formularios, campos especiales en formularios, y sigue una larga lista de funcionalidades.

HTML permite incrustar código JavaScript y hojas de estilo en el mismo documento, o en fuentes externas. Además, existen en la web múltiples herramientas que ayudan al programador con plantillas y código autogenerdo.

2.4.5 CSS

Cascading Style Sheets (CSS) es un estándar para la definición de estilos, esto permite al programador diseñar y decorar páginas webs. Es una herramienta necesaria ya que facilita enormemente la colocación de elementos, el coloreado y presentación del estilo de la web. Básicamente convierte texto plano e imágenes en algo atractivo para la captar la atención del usuario.

2.4.6 R

Para establecer conexión con el entorno de trabajo es necesaria una API que ofrezca funciones de procesado de datos cuando la aplicación lo necesite. Con éste propósito, R ofrece varios paquetes que ayudan al desarrollo de un servidor HTTP en R, como *Rook*, *Shiny*, o *OpenCPU* como lo que permite la realización de operaciones sobre un dataset.

2.4.6.1 *Rook*

Se trata de un paquete de R desarrollado por Jeffrey Horner, que crea una interfaz para las funciones de R HTTP server. Este paquete abstae un servidor web a varios objetos y métodos que son llamados del mismo modo que otros objetos o métodos en R, como consecuencia crear un servidor en R con Rook resulta simple ya que solo se necesita un script en R con llamadas a funciones de Rook.

Otra ventaja de Rook es que permite escribir respuestas HTTP para cada petición y personalizar las cabeceras, lo que ofrece mayor versatilidad cuando se programan servicios web para un servidor.

2.4.6.2 *Shiny*

Shiny es un paquete de RStudio preparado para desplegar un servidor web para R. Está provisto de funciones que para el desarrollo de páginas web lo que permite al usuario realizar operaciones en R y lanzar los resultados con elementos gráficos en una interfaz mejorada visualmente.

El problema que plantea Shiny es que es un paquete creado para el desarrollo de páginas web y como consecuente no está preparado para desplegar una API REST. No hay código HTML solo funciones de Shiny, lo que condiciona críticamente la presentación del contenido web.

2.4.6.3 *OpenCPU*

Ésta librería de R ofrece herramientas para el desarrollo de un servidor en la nube con servicios de computación estadística y compatibilidad para crear interfaces HTTP. La API HTTP ofrece soporte para la maoría de métodos HTTP. Además, proporciona integración, y tanto sencillos métodos de entrada y salida como simples tipos de datos.

La desventaja de OpenCPU es que no permite la escritura de las respuestas directamente, lo que limita al programador a los métodos proporcionado lo que puede no ser útil para su objetivo.

2.4.7 CartoDB

Para la representación de datos sobre mapa, cartoDB nos ofrece una plataforma abierta, intuitiva y escalable para geolocalizar datos, además de permitirnos manejarlos ya que posee un entorno de SQL para dicho propósito. Además, cartoDB cuenta con múltiples tipos de representación entre los que incluye representaciones “in time” por lo que se pueden recrear escenas con datos a lo largo de un tiempo.

2.4.8 Google Maps

Tratando datos geo posicionados no podemos pasar por alto Google Maps, siendo una de las herramientas más populares. Proporciona una API sencilla e intuitiva con múltiples funcionalidades, entre las que, como CartoDB proporciona representación de datos “in time” y presenta una gran compatibilidad web y para sistemas operativos como Android o iOS, puesto que este es su uso principal.

2.5 Algoritmos de análisis

La mayor parte de las tecnologías involucradas en la minería de datos son algoritmos utilizados para clasificar, puntuar o separar palabras del dataset. Dichos algoritmos son bien conocidos, por lo que están lo suficientemente desarrollados como para proporcionar un mayor rendimiento que el de un algoritmo propio.

Éste es el punto por el que R es fundamental para el desarrollo de este proyecto, las funcionalidades básicas de minería de datos son soportadas por R. Adicionalmente R posee librerías que mejoran en gran medida éstas capacidades. Como es el caso de stats, plyr o tm; que son librerías que han de ser añadidas a nuestro entorno.

La librería Stats proporciona herramientas para el análisis avanzado de datos, como el algoritmo k-means [6]. Plyr ofrece distintas posibilidades para la computación en paralelo de grandes cantidades de datos. Tm nos proporciona estructuras y algoritmos optimizados para la minería de datos. A parte de funciones para crearlos. A nivel de presentación gráfica contamos con librerías como wordcloud [7] y diferentes versiones de gplot [8].

2.5.1 TF-IDF

El posicionamiento de las palabras es un punto que es necesario cubrir, para ello los algoritmos incluyen modelos matemáticos que tratan de puntuar y discriminar palabras, para ello obtener las más relevantes, dependiendo de diversos factores, teniendo en cuenta tanto el documento en sí como toda la colección.

Para dicho posicionamiento los algoritmos toman como parámetro indispensable la frecuencia de término en el documento y en el conjunto de ellos. Una vez obtenidas dichas frecuencias es menos costoso a nivel computacional aplicar algoritmos de puntuación y los resultados son más fiables, un ejemplo de esta preparación es la Ilustración 2: Filtrado de un dataframe para la aplicación de tf-idf.

```
df<-as.data.frame(finalDF)
s.corpus = Corpus(VectorSource(df))
sclean.corpus = tm_map(s.corpus,content_transformer(tolower))
sclean.corpus = tm_map(sclean.corpus,removeWords,c(stopwords("english"), stopwords
sclean.corpus = tm_map(sclean.corpus,removePunctuation)
sclean.corpus = tm_map(sclean.corpus,stripWhitespace)
dtm<-DocumentTermMatrix(sclean.corpus)
terms<-removeSparseTerms(dtm, 0.9)
```

Ilustración 2: Filtrado de un dataframe para la aplicación de tf-idf

El algoritmo más relevante para este apartado es Term Frequency Inverse Document Frequency (TF-IDF). Éste algoritmo es utilizado principalmente para extraer las palabras clave dentro de una gran colección de textos. El algoritmo TF-IDF computa la nota TF-IDF para cada término y documento. El valor TF- IDF aumenta proporcionalmente al número de veces que una palabra aparece en cada documento, pero se compensa con la frecuencia de la palabra en la colección, con el fin de evitar la confusión con palabras comunes que aportan menos información para el estudio.

Matemáticamente TF-IDF es el producto de la frecuencia de término y la frecuencia inversa del documento, donde la frecuencia de término (tf) es el número de

veces que un término “a” ocurre en un documento “x”, esto es conocido como la frecuencia bruta de término f(t,d) en comparación al total del documento.

$$\mathrm{tf}(t,d) = \frac{\mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}}$$

Ilustración 3:Formula TF

Para calcular la frecuencia inversa del documento, dividiremos el número total de documentos |D| entre el número de documentos que contienen el término, tomando el logaritmo como resultado.

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Ilustración 4: Fórmula de IDF

El cálculo de TF-IDF se obtiene del producto de los factores previamente calculados, la conclusión es que un valor alto de tfidf se obtendrá si un término tiene una frecuencia muy alta en un documento, pero una frecuencia baja en el total de documentos.

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,D)$$

Ilustración 5: Fórmula de TF-IDF

2.5.2 K-means

Se trata de un método de agrupamiento o clustering que busca la partición de un grupo de observaciones en K grupos, en el que cada observación pertenece al grupo con valor medio más cercano. La idea es definir K centros, uno por cluster, dichos centros han de ser situados de manera estratégica, ya que diferentes localizaciones generan diferentes resultados, generalmente se sitúan lo más lejos posible entre ellos.

Una vez definidos los centros, se asocian el resto de los puntos al centro más cercano, una vez agrupados todos los puntos se han de recolocar los k centros. Para ello se calculan los nuevos centros como el baricentro [9] de los cluster resultantes del paso anterior. Una vez calculados los nuevos centros, se vuelven a unir los puntos por proximidad creando nuevos clusters. Estos dos últimos pasos se repiten de manera iterativa hasta que el algoritmo converja.

Las ventajas principales del algoritmo son su rapidez y robustez, dando una eficiencia relativamente buena, mejorando considerablemente cuando los núcleos de los clusters son apreciables o distantes.

Por otro lado, el algoritmo necesita un número específico de clusters para funcionar, lo cual no siempre es posible además de que cuando los datos se sobreponen el algoritmo puede no distinguir entre los clusters sobrepuestos, esta última condición es muy relevante a la hora de tratar datos con ruido, ya que es prácticamente imposible que los datos no se sobrepongan.

2.5.3 Modelos lineales

Los modelos lineales intentan explicar el comportamiento de una variable aleatoria mediante su relación lineal con los valores de otras que puedan ser inluyentes

En el caso de multiples variables, se conoce como regresión lineal múltiple al análisis del comportamiento de una variable "Y" a través de otro conjunto de variables "X1…Xk" y un término aleatorio ε. Para poder aplicar un modelo lineal se requiere de los siguientes pasos:

- Modelo: planteamiento y definición de las variables que intervienen
- Muestra aleatoria: número de observaciones que van a realizarse y procedimiento a seguir (modelo teórico del training set).
- Datos: Valores numéricos obtenidos de las observaciones previstas.
- Aplicaciones de las técnicas estadísticas a otro conjunto de datos, o testing set.

La fórmula que descrie un modelo lineal es:

$$Y_t = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$$

Donde:

- Y es la variable dependiente, la que se espera estimar.
- X1...Xn son las variables independientes o regresores
- B_0…βn son los parámetros que miden la influencia que las variables explicativas tienen sobre el parámetro estimado.
- B_0 es la intersección o termino constante.

3 Metodología

3.1 Crawling

La extracción de datos de la web de “El Tenedor” [10] se compone de dos módulos, un programa de extracción de contenido web y otro para parsear la información útil, ambos programados en Python.

3.1.1 Módulo de extracción de contenido html

Es posible obtener la información de la base de datos de manera iterativa, ya que utilizan para las peticiones al servidor el método GET [11] de HTTP [12] , al utilizar este método la información del envío del formulario HTTP se puede ver en claro en la URL, por lo que en la mayor parte de los casos son deducibles, y podemos obtener toda la información simulando peticiones.

Para obtener los datos de una manera eficiente, necesitamos un algoritmo que nos permita obtener una gran cantidad de contenido html de manera mecanizada, esto es conocido como web crawler o araña web. La base del procedimiento es acceder a los enlaces de la página web inspeccionada de manera iterativa y organizada.

El acceso a las URLs viene determinado por el diseño de la página web, por lo que podemos tomar ventaja de las clases de CSS, o de los propios elementos de html para localizar los puntos de cada página de los que podemos obtener algún tipo de beneficio, para ello utilizaremos la librería de BeautifulSoup [13], esta librería nos permite parsear el documento descargado (parser xml), o también encontrar coincidencias con expresiones regulares. De este modo buscaremos todos los redireccionamientos a páginas de las que nos interese descargar el código.

Al realizar consultas multiples a un servidor, surgen ciertos problemas puesto que es habitual recibir rechazo a peticiones masivas con el mismo origen y no siendo un explorador web. Para ello utilizaremos funciones de la librería pyCurl [14], éstas nos permitirán simular un explorador web, en éste caso Mozilla Firefox, y haremos esperas activas aleatorias entre peticiones a diferentes URLs estableciendo una cabecera USER-AGENT para cumplir con el standard de la petición.

El contenido descargado será organizado de la siguiente manera por comodidad de acceso:

→Raíz
→DatosMadrid
→Alcalá de Henares
->Restaurantes.html
→Getafe
->Restaurantes.html
…
→DatosCataluña
→Girona
->Restaurantes.html
…

3.1.2 Módulo de parseo de contenido

Para filtrar un documento html, procederemos a abrir uno de los documentos ya descargados y realizar en análisis sintáctico con BeautifulSoup, utilizaremos el formato "lxml", ya que es compatible con el contenido y Windows 7.

Una vez obtenido un objeto Soup, es muy fácil extraer el contenido de una clase expecífica con la función find(tag, exp) [13], o directamente extraer una lista con todos las coincidencias con find_all(tag,exp) [13]. El filtro dependiendo del caso se hará por la clase del elemento html, id, expresión regular, etc. Para acceder al contenido de texto no todo podremos hacerlo con la librería Beautifulsoup, por lo que en algunos casos tendremos que utilizar expresiones regulares para realizar el análisis sintáctico con la librería re [15].

```
f = open(item, "r")
soup = BeautifulSoup(f, "lxml")
rate = soup.find_all("span", class_="reviewItem-ribbon rating")
comment = soup.find_all('div', class_= "reviewItem-customerComment")
maestry = soup.find_all('span',class_="reviewItem-profileGamification")

try:
    while(1):
        userRate = rate[i].span.string
        userMaestry = maestry[i].span.string
        userComment = comment[i].string
        userComments.append(userComment)
        userMaestries.append(userMaestry)
        userRates.append(userRate)
```

Ilustración 6 fragmento de Código para la extracción de tres variables

```
try:
    regex = re.compile("mboxRecs: ({.*})")
    magicItem = re.findall(regex, unicode(soup))

    magicItemx = json.loads(unicode(magicItem[0]))
    restaurant.update(magicItemx)

except:
    restaurant.update({"InfoBox": "none"})
```

Ilustración 7 Fragmento de código para la extracción con expresiones regulares

Toda la información no está clasificada en tags definidos por el código HTML, una gran parte de la información relevante se encuentra bien en la cabecera de la página, como parte del código JavaScript incrustado, normalmente con la estructura del formato JSON, o bien en zonas del documento donde hay que filtrar parte del texto ofrecido dentro de un tag.

Para ello la librería re de Python nos ofrece una solución que requiere más trabajo por parte del programador, ya que para cada variable o conjunto de ellas se ha de diseñar la expresión o expresiones regulares deseadas, como es el ejemplo de la expresión de la Ilustración 3.

Una vez diseñada la expresión regular la función compile(string) guardará en caso de ser correcta la sintaxis de dicha expresión, que más tarde será usada por la función findall(regex,string), que como en el caso de BeautifulSoup, nos devolverá una lista con todas las coincidencias en el documento HTML.

Ya obtenidas todas las variables descritas en el punto 4.1.1 Descripción del dataset se han de guardar en la base de datos para que puedan ser fácilmente accesibles, para ello nos aprovecharemos de que Python tiene diccionarios cuya sintaxis es igual a un archivo con formato JSON, que es fácilmente importable por mongoDB.

```
restaurant.update({"comment":userComments,
"maestry":userMaestries,
"rate": userRates,
"_id" : _id,
"zone": zone})
```

Ilustración 8 Inserción de datos en un diccionario Python

Una vez obtenido el diccionario, ya solo queda introducirlo a la base de datos, éste paso es muy simple con Python por la compa-tibilidad de formatos y la librería pyMongo [16]. Para ello necesitaremos crear una conexión cliente-servidor con la BBDD, definiendo un cliente con la función mongoClient() tendremos acceso ella. Una vez conectados, accederemos a la colección pertinente Madrid/Cataluña e insertaremos el nuevo restaurante con la función collection.insert_one(Python diccionary).

4 Análisis de restaurantes

4.1 Analítica descriptiva

4.1.1 Descripción del dataset

El dataset se divide en dos grupos que contienen la información de los restaurantes de Madrid y Barcelona respectivamente.

La cantidad de datos obtenida para el análisis es de 1,96 GB para Madrid, con 1903 restaurantes y de 1,29 GB para Cataluña con 2377 restaurantes.

Tras un exhaustivo parseo y limpieza de la información obtenida, obtenemos los datos útiles para el estudio, en formato JSON, siendo el peso de estos datos 92,3 MB para Madrid y 63.6 MB para Cataluña.

La cantidad de datos obtenidos, no será igual a la de los datos útiles, por lo que para lograr un objetivo con cada estudio, el dataset quedó dividido de la siguiente forma:

Variable	Tipo	Descripción
Id	numeric	Identificador para la base de datos
restaurantID	numeric	Identificador en el Tenedor
categoryID	numeric	Identificador de categoría
categoryName	string	Nombre de la categoría
slug	string	Nombre del restaurante
pageUrl	string	Link a la página del restaurante
thumbnailUrl	string	Link a la imagen del restaurante
value	numeric	Precio medio del restaurante
city	string	Ciudad
salePrice	numeric	Precio despues de aplicar oferta
rating	numeric	Nota media del restaurante
foodRating	numeric	Nota media de la comida
serviceRating	numeric	Nota media del servicio
ambianceRating	numeric	Nota media del ambiente/local
qualityPriceRating	numeric	Nota media de la calidad/precio
saleTypeId	numeric	Identificador de la oferta
saleTypeName	string	Descripción de la oferta
saleTypeIsMenu	boolean	Oferta de menú
saleTypeIsSpecialOffer	boolean	Oferta por grupos/ individual
saleTypeMinPartySize	numeric	Personas míminas para aplicar oferta
saleTypeMaxPartySize	numeric	Máximo de personas para aplicar oferta
isRecommendable	boolean	Recomendador de elTenedor
Lat	numeric	latitud de la posición del restaurante
Lng	numeric	longitud de la posición del restaurante
MultiTag	list of strings	lista de tags por restaurante
Card	list of strings	carta del restaurante, puede incluir precio
comment	list of strings	lista de comentarios de usuario por restaurante
Maestry	list of strings	nivel de experiencia de cada usuario que comenta
Rate	list of numerics	nota de cada usuario al restaurante

Ilustración 9: Tabla de variables obtenidas

4.1.2 Estudio de tags

4.1.2.1 Dataset:

Para lograr un Dataset con el que poder operar en r, nos vemos en la necesidad de tratar los datos obtenidos, para ello haremos uso de las funciones más adecuadas en R, optimizando la eficacia del algoritmo, tales como la función lapply [17], esta función nos permite recorrer una lista o vector aplicando una función a cada elemento de ella, retornando como resultado una lista con la misma longitud que la pasada por parámetro.

```
tag = lapply(find_all,function(x){
col2 <- x$multiTag
return(col2)
})
untag<-unlist(tag)
topie<-count(untag)
```

Ilustración 10 Ejemplo básico de selección del dataset general

Tras obtener los datos que queremos del dataset, los ordenamos, observamos su distribución y eliminamos marginales y resultados inválidos tales como algunos tags de localización generales que no aportan información o tags de frecuencia despreciable.

x	freq
Con amigos	1363
Tradicional	1171
Familia	1081
MediterrÃ¡neo	771
Moderno	741
De negocios	593
CumpleaÃ±os	586

Ilustración 11 Ejemplo del dataset de tags

Podemos clasificar los tags en tres subgrupos, en el primero identificamos los tags por categoría de restaurante, es decir dependiendo del tipo de comida que sirvan, ejemplo de esto son los restaurantes tradicionales, modernos, de carnes, italianos, asiáticos, etc... Por otro lado, observamos un subconjunto de datos referentes a zonas geográficas como mediterráneo, malasaña o la rambla. Por último existe un conjunto de tags que hacen referencia a eventos, los clasificaremos como tags sociales dentro de este grupo entran tags como con amigos, de negocios o bodas.

4.1.2.2 Representación gráfica de tags

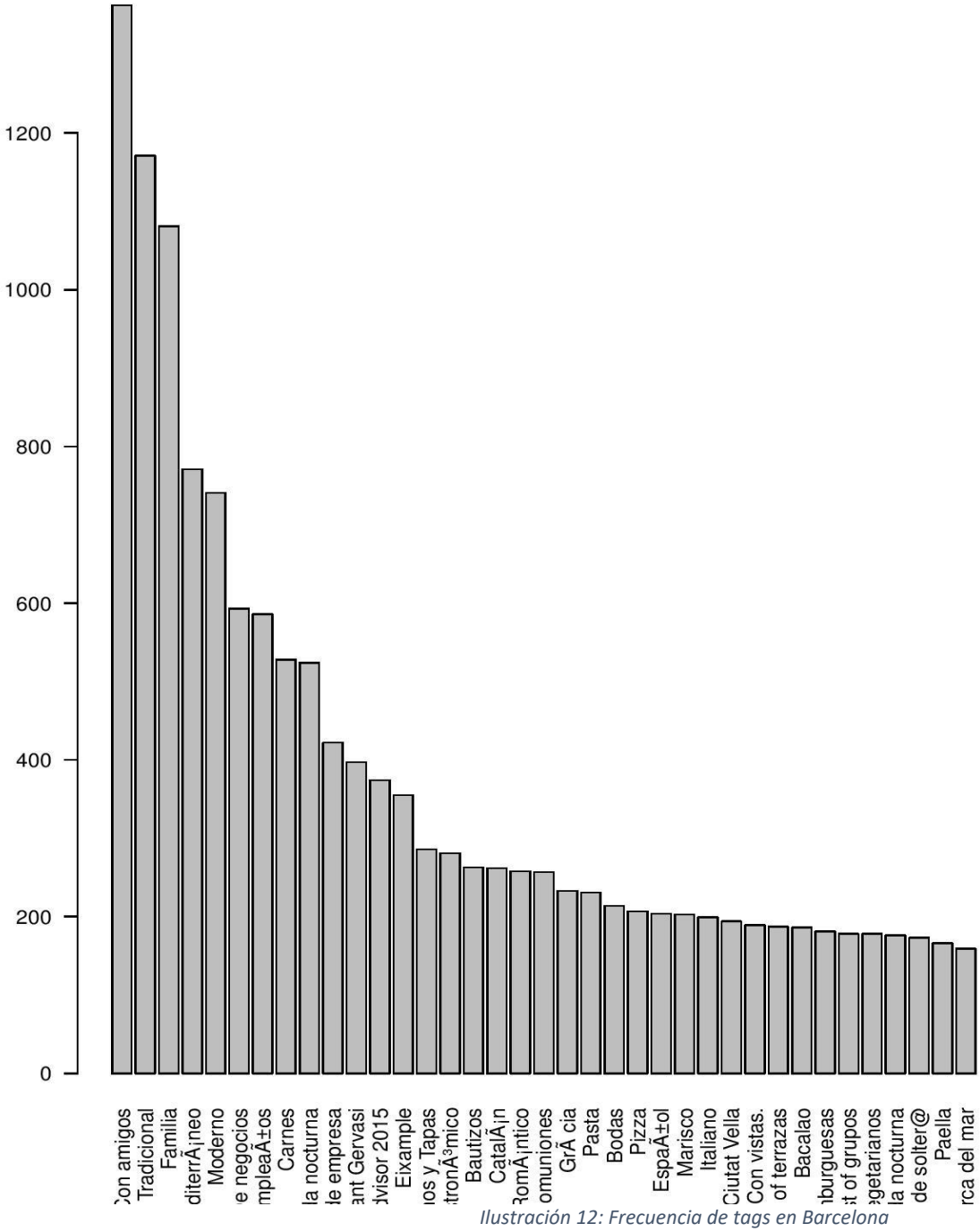


Ilustración 12: Frecuencia de tags en Barcelona

En la Ilustración 12: Frecuencia de tags en Barcelona podemos observar a primera vista una diferencia en la tendencia de los tags, existen unos valores de pico como son los tags “Tradicional” y “Familia” y apreciamos un descenso

gradual de la frecuencia hasta el tag “Pasta”, donde comienza a estabilizarse y todos los tags a partir de ahí rondan la frecuencia 200. Analizando estos diferentes tramos por partes, observamos que los tags sociales tales como “Con Amigos” o “Familia”, toman el liderazgo, dentro de la categoría gastronómicas los tags “Tradicional” y “Moderno” se alejan de los siguientes ”catalán”, “Pasta”, “Pizza”, ”Italiano”, etc… por 450 unidades. Dentro del tipo de tag geográfico en este caso tan solo encontramos uno al final de la lista, el tag “cerca del mar”.

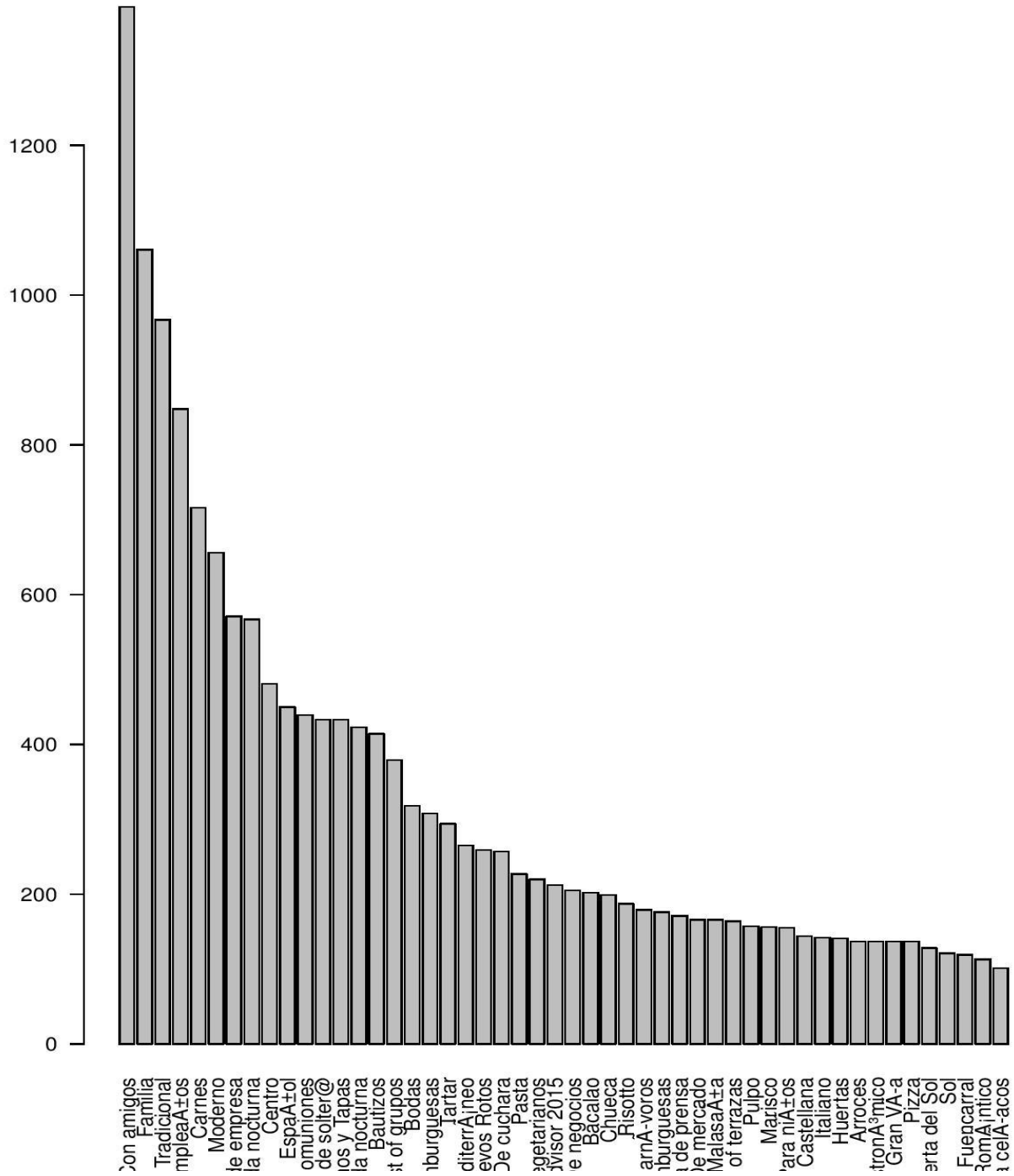


Ilustración 13 Frecuencia de tags en Madrid

En la Ilustración 13 Frecuencia de tags en Madrid, podemos observar un comportamiento similar a Barcelona, siendo el top 3 el mismo, aunque en diferente orden. A la hora de comparar los tags sociales siguen líderes “Con amigos” y “familia”, este último tiene mayor frecuencia en Madrid, incluso teniendo una menor cantidad de restaurantes en la base de datos. Las diferencias aparecen cuando tenemos en cuenta los tags de categorías gastronómicas, si bien “Tradicional” lidera de nuevo la lista, hay otros como “Carnes” o “Español”. Como curiosidad, podemos observar que este tag no está presente en la capital catalana y se ve sustituido por el tag “Catalán”. En las siguientes posiciones donde la frecuencia comienza a estabilizarse, de nuevo a la altura de “Pasta” se ven con igual asiduidad los tags referentes a restaurante italiano como “italiano”, “risotto”, ”pizza”, etc.. Analizando por último los tags de posición, podemos observar que el más frecuente es “Centro”, si tenemos en cuenta el mapa de densidad de la Ilustración 18: Distribución de los datos en Madrid es obvia la deducción que es donde mayor cantidad de restaurantes, seguido por barrios de la capital, como son “Malasaña” y “Huertas”, estos últimos tags no son zonas al azar, existe una mayor frecuencia en éstos porque son zonas que “venden” ya que son, o han sido hasta la fecha consideradas zonas de moda entre la población joven, que son los principales consumidores en portales web.

Como conclusión, los tags más populares en ambos sitios coinciden, siendo éstos, “Con amigos”, “Familia”, “Tradicional”. Los dos primeros son genéricos y la información que dan es previsible, un restaurante es, a priori un entorno social.

Analizando los tags de popularidad alta sin ser el top tres, descubrimos que la zona de Cataluña se ve altamente influenciada por la costa y el turismo, con tags como “Cerca del mar”, “Paella”, “Vida nocturna” o “Español”. En Madrid podemos observar que podría existir una mayor diversidad gastronómica, ya que el número d tags referentes a diferentes tipos de comida es mucho mayor, también hemos de destacar que el centro es una zona turística, por esto el tag “Centro” aparee en la cabeza junto a “Vida nocturna” o “Best of grupos”.

Por último cabe destacar que ambas zonas tienen un gran peso económico en España por lo que no pueden faltar indicadores como “Cena de Empresa” o “Cena de Negocios”.

4.2 Comparación cuantificable de variables

Relación calidad- precio (Cataluña + Madrid):

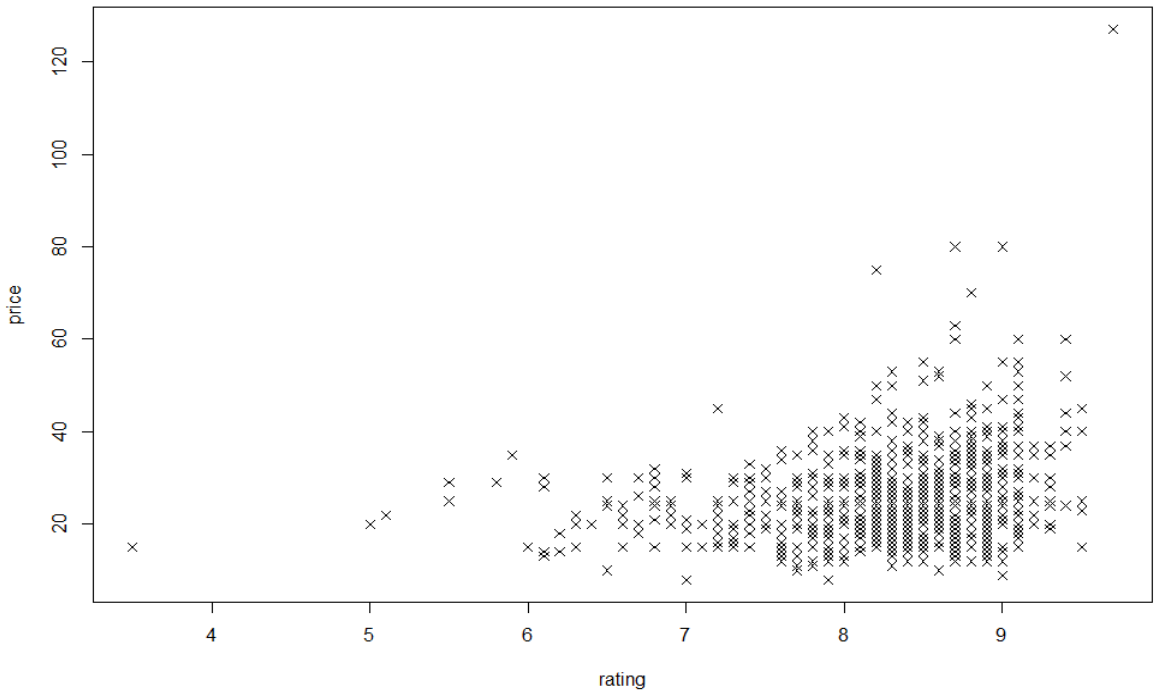


Ilustración 14 Representación de restaurantes en su relación nota-precio

Suponiendo la hipótesis inicial de que la calidad tiene un precio, podemos observar que se cumple parcialmente, puesto que disponemos de una gran variedad de precios en restaurantes de una calidad similar, cabe destacar que los peores restaurantes tienen en común un precio bastante bajo.

En el marginal superior de la gráfica vemos que el mejor evaluado de todos, se corresponde con el de mayor precio, además, observamos que no existe ningún restaurante caro, que no cumpla un criterio mínimo de calidad, es decir todos los restaurantes de precio alto tienen una buena recepción por los clientes.

La conclusión es que no hay que pagar mucho para comer bien en éstas comunidades, ya que existe una oferta bastante amplia en los precios medios/bajos, aunque los restaurantes excelentes sí que suponen un mayor coste.

4.2.1 Boxplot

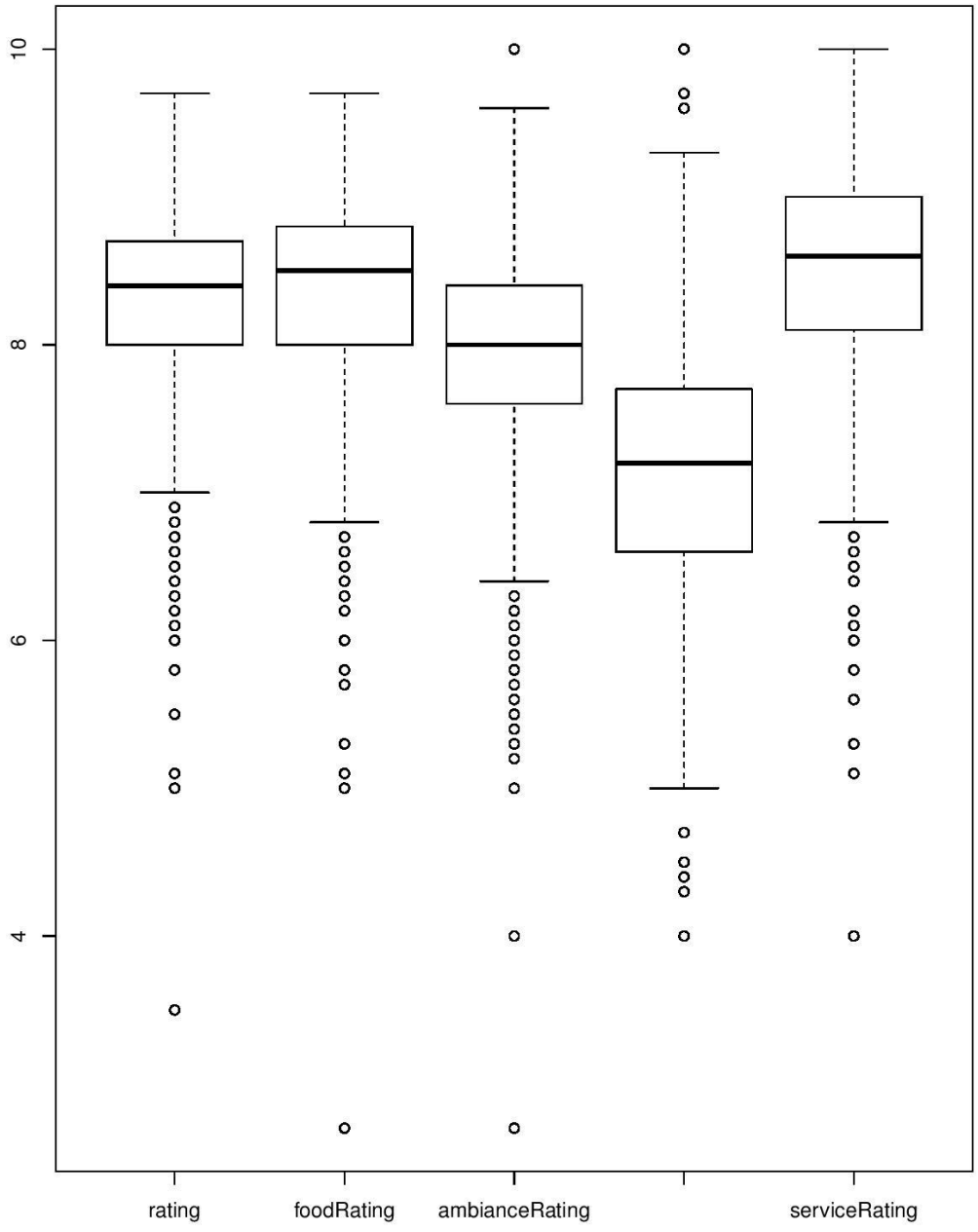


Ilustración 15 Boxplot de distribucion de valoraciones de los restaurantes

price	rating	foodRating	ambianceRating	qualityPrice	serviceRating
25.03696	8.292608	8.372895	7.900342	7.176044	8.479466

Ilustración 16: Medias de ratings excluyendo marginales

Analizando punto por punto, observamos que la nota del servicio es superior a las demás, con lo que una conclusión podría ser que en España se valora el trato personal de manera positiva, lo mismo sucede con la comida, por norma general se come bien en los restaurantes españoles y el ambiente es adecuado, aunque este por debajo de la media del servicio.

Podemos tomar como conclusión que la gente establece la calificación, cuya media varía entre el 7 y el 8,5, siendo la media de 7 la más alejada de la media general de "rating" situada en 8,2, por lo que podemos deducir que la gente es más crítica con el dinero que paga por la comida, que por la comida en sí misma, el precio es un factor más sensible y se ve una mayor desconformidad del consumidor.

Fundamentalmente, podemos decir que la gente tiende a ser muy positiva con las notas en general ya que incluso la menor nota media es un 7/10.

4.2.2 Scatterplots

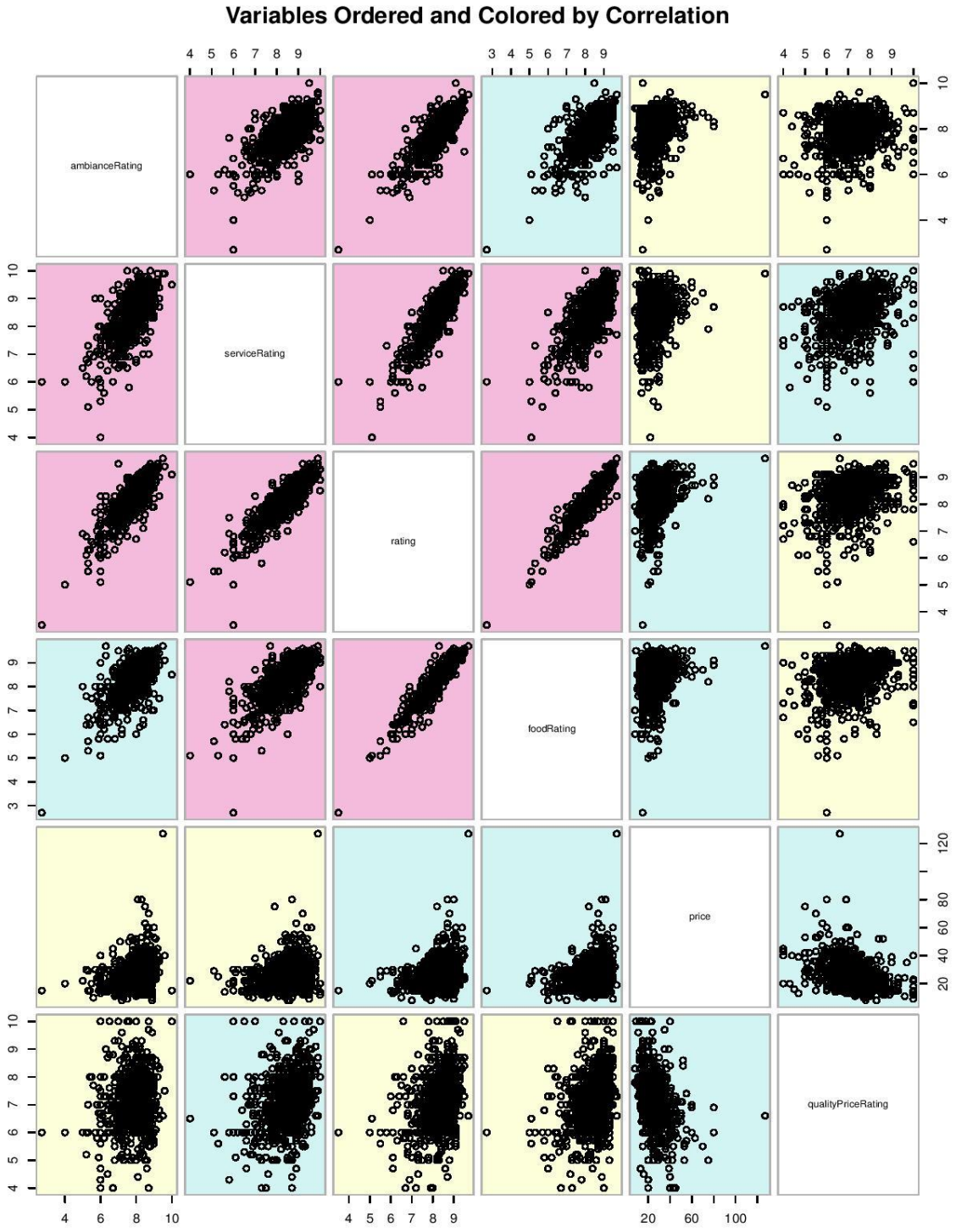


Ilustración 17: Scatterplot de variables calidad y precio

La Ilustración 17: Scatterplot de variables calidad y precio representa todas las variables referentes comparándolas entre sí y ordenando las imágenes por

niveles de correlación, así se verán representadas desde rojo las que tengan un mayor grado de correlación a verde, con una correlación baja.

Analizando el scatterplot por áreas podemos observar que en el cuadrante superior izquierdo existe una zona donde hay un alto grado de correlación, es decir las variables ambianceRating, serviceRating, raing y foodRating están altamente relacionadas entre sí como era de esperar, ya que cuando representamos las notas en boxplots, véase la Ilustración 15 Boxplot de distribucion de valoraciones de los restaurantes, las distribuciónes de las diferentes variables de nota eran similares.

Las diferencias comienzan cuando analizamos la variable precio frente a las demás, podemos observar que está bastante correlada a la calidad de la comida, seguidas por la calidad del servicio y el ambiente, lo que posee una correlación un poco menor. Por lo que podemos deducir que se paga más acorde a la calidad de la comida que a la ambientación del restaurante en sí.

Si comparamos las variables precio y calidad-precio, vemos que existe una correlación baja, teniendo en cuenta la distribución de restaurantes por precio, existe una gran variedad de restaurantes baratos, con notas altas y muy pocos con notas bajas, además los restaurantes caros también se corresponden con notas altas, por lo que la conclusión de la comparación de estas variables es que no son apenas dependientes.

5 Representación por geolocalización

5.1 Comunidad de Madrid

5.1.1 Distribución de restaurantes por densidad

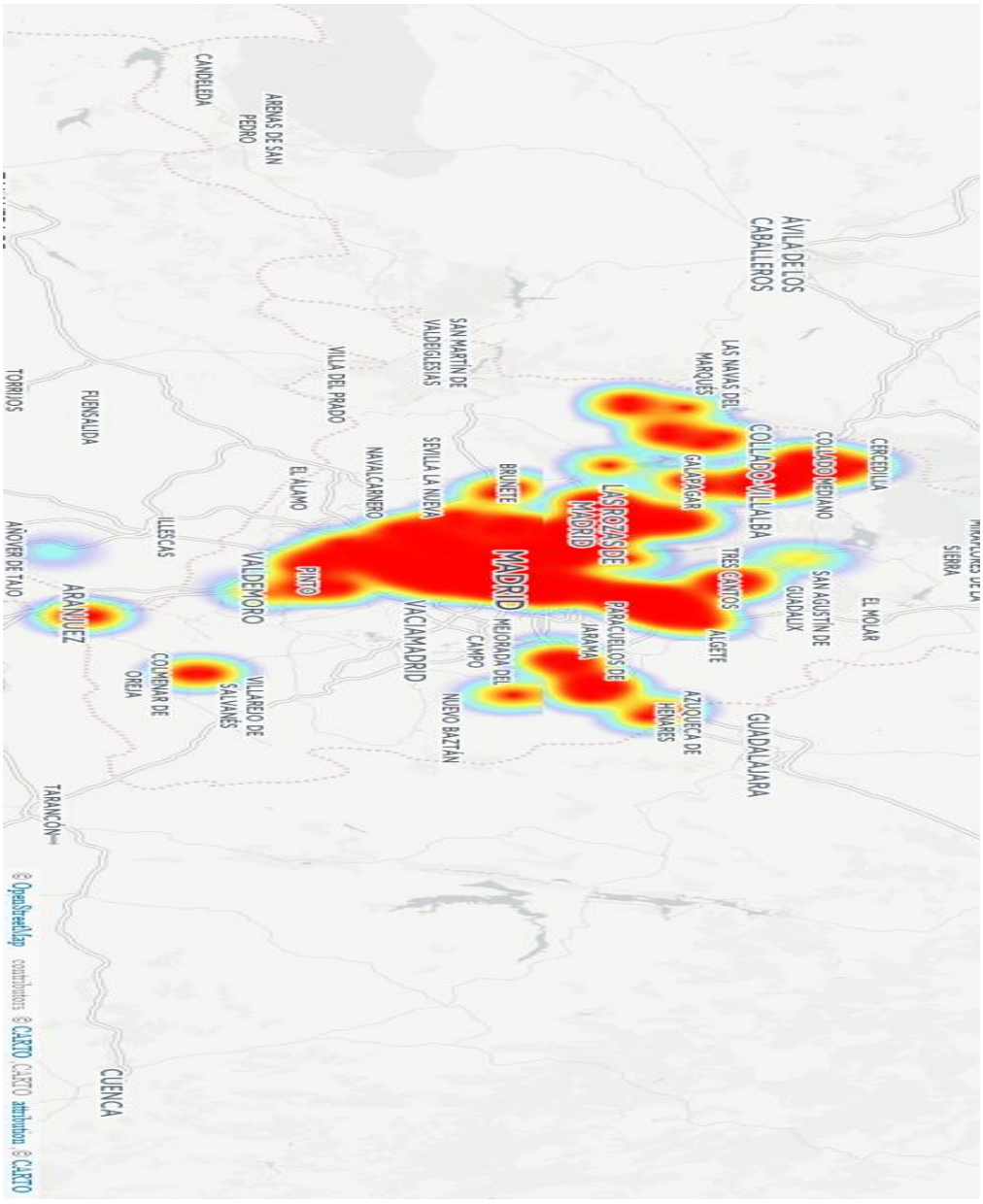


Ilustración 18: Distribución de los datos en Madrid

El mapa de densidad de la Ilustración 18: Distribución de los datos en Madrid nos muestra cómo están distribuidos los restaurantes sobre la comunidad madrileña y,

podemos determinar que la mayor parte de ellos se encuentran en la zona centro, aunque se distinguen también un núcleo en la zona norte a la altura de Collado-Villalba, y zonas de menor densidad al sur y este de Madrid.

5.1.2 Distribución de restaurantes en función del precio medio

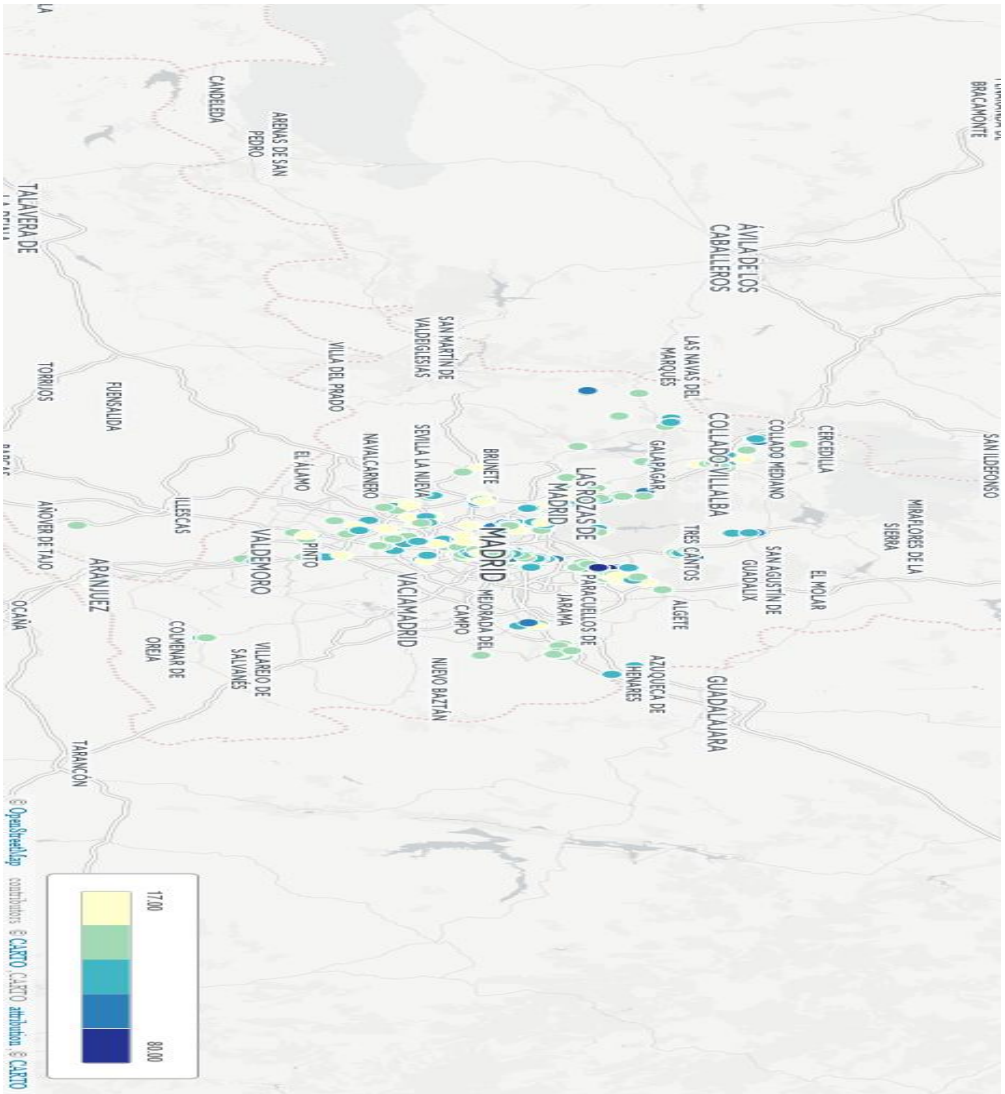


Ilustración 19: Distribución de restaurantes en función del precio medio

La Ilustración 19: Distribución de restaurantes en función del precio medio nos indica que la variación de precio entre el restaurante más barato y el más caro es relativamente baja, siendo el precio menor bastante elevado aunque, se ha de tener en cuenta que es el precio medio por carta, se excluyen menús y otras promociones. Debido a la alta densidad de restaurantes, no podemos obtener una conclusión a primera vista, por lo que separaremos el dataset en tres, precio medio, alto y bajo.

```
> summary(tocartoi$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8.00   18.00   24.00   24.89   30.00   80.00
```

Ilustración 20: resumen de precios en Madrid

Para el criterio de separación tendremos en cuenta la información ofrecida por la Ilustración 20: resumen de precios en Madrid , realizando una separación por cuartiles donde consideraremos baratos los restaurantes inferiores a 18 euros, y caros los que superen los 30.

5.1.3 Distribución de restaurantes con precio medio bajo

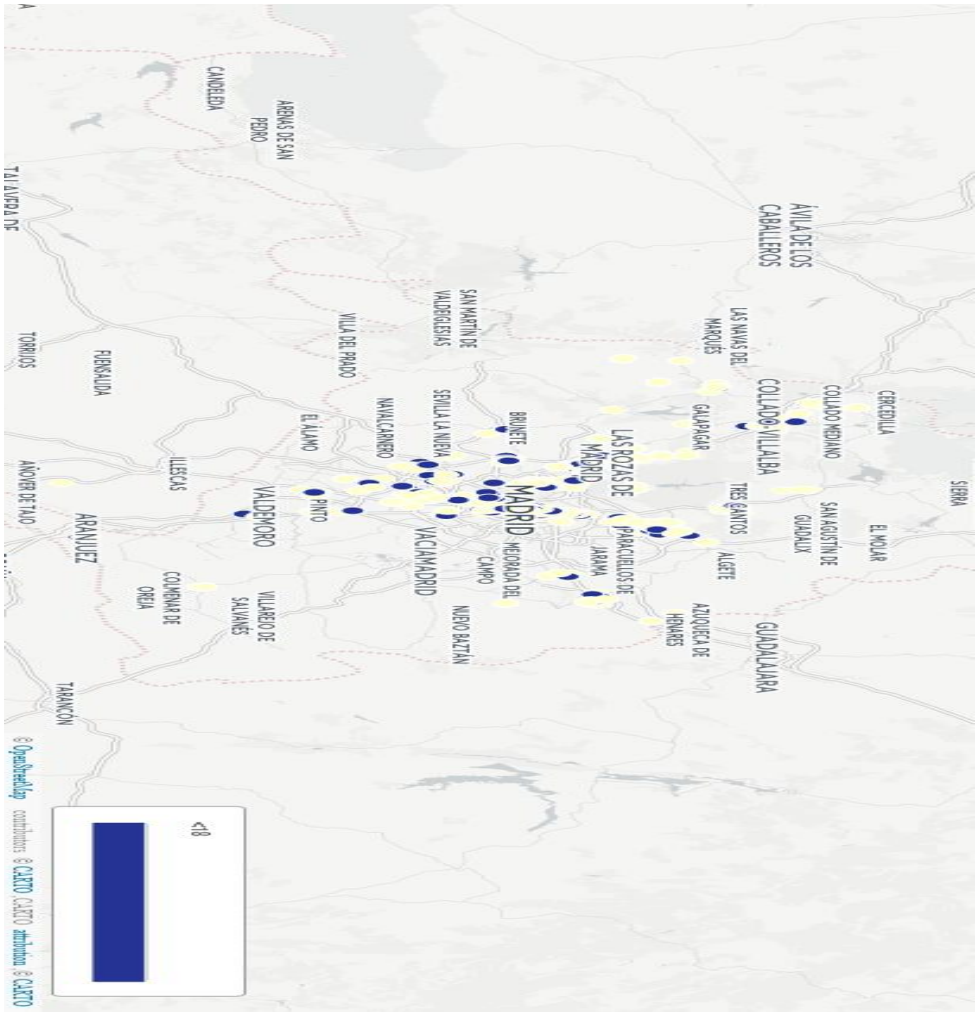


Ilustración 21: Distribución de restaurantes con precio medio bajo

Si trazamos una línea horizontal a la altura del título Madrid, teniendo en cuenta que es el punto central, podremos hacer una diferenciación entre la zona norte y zona sur. En la zona norte podemos apreciar una menor cantidad de restaurantes baratos, mientras que en la zona centro-sur y sur hay núcleos de restaurantes baratos en prácticamente todas las ciudades.

5.1.4 Distribución de restaurantes con precio medio intermedio

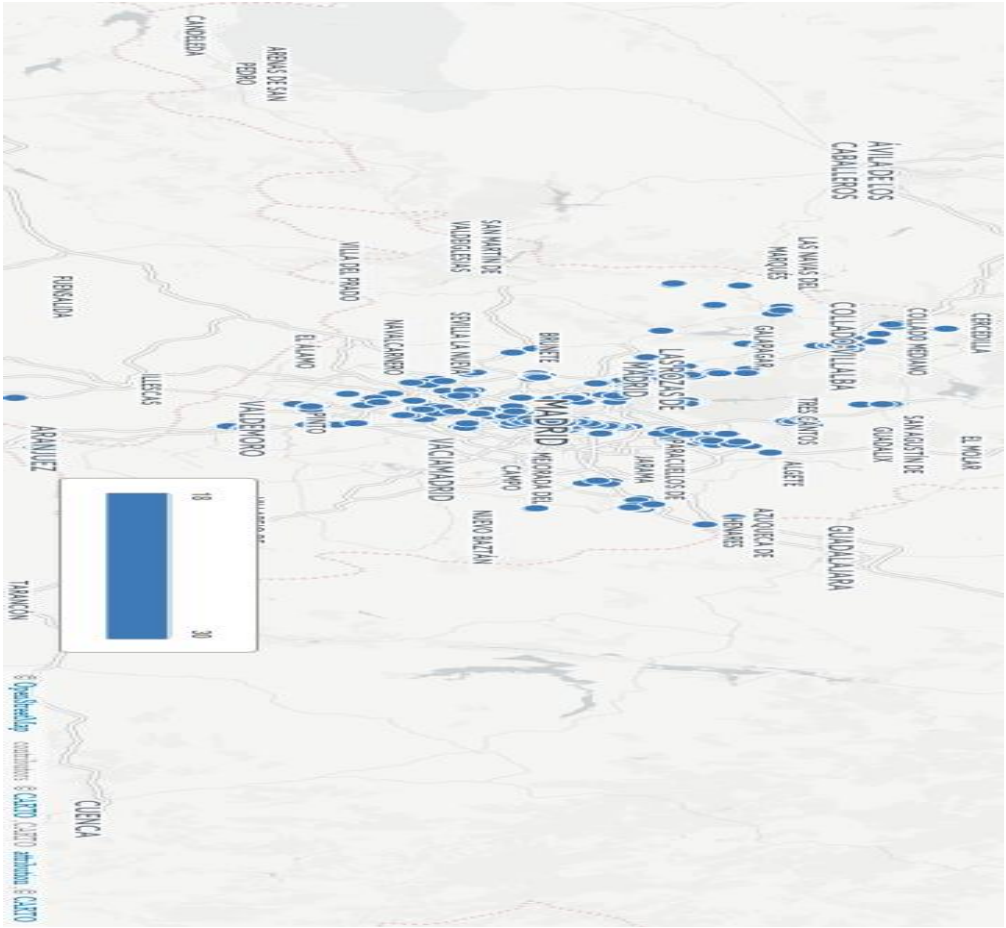


Ilustración 22: Distribución de restaurantes con precio medio intermedio

5.1.5 Distribución de restaurantes con precio medio alto

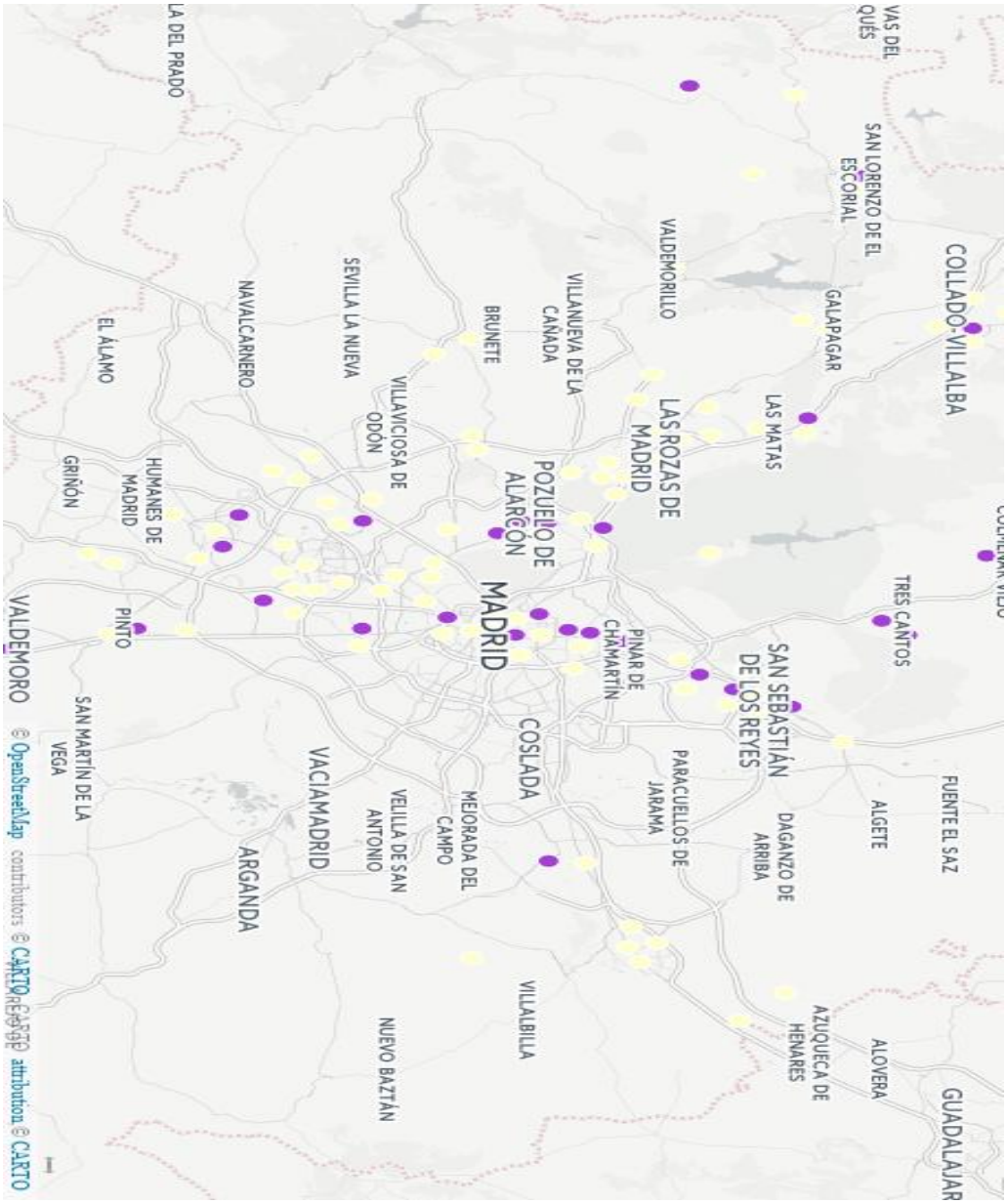
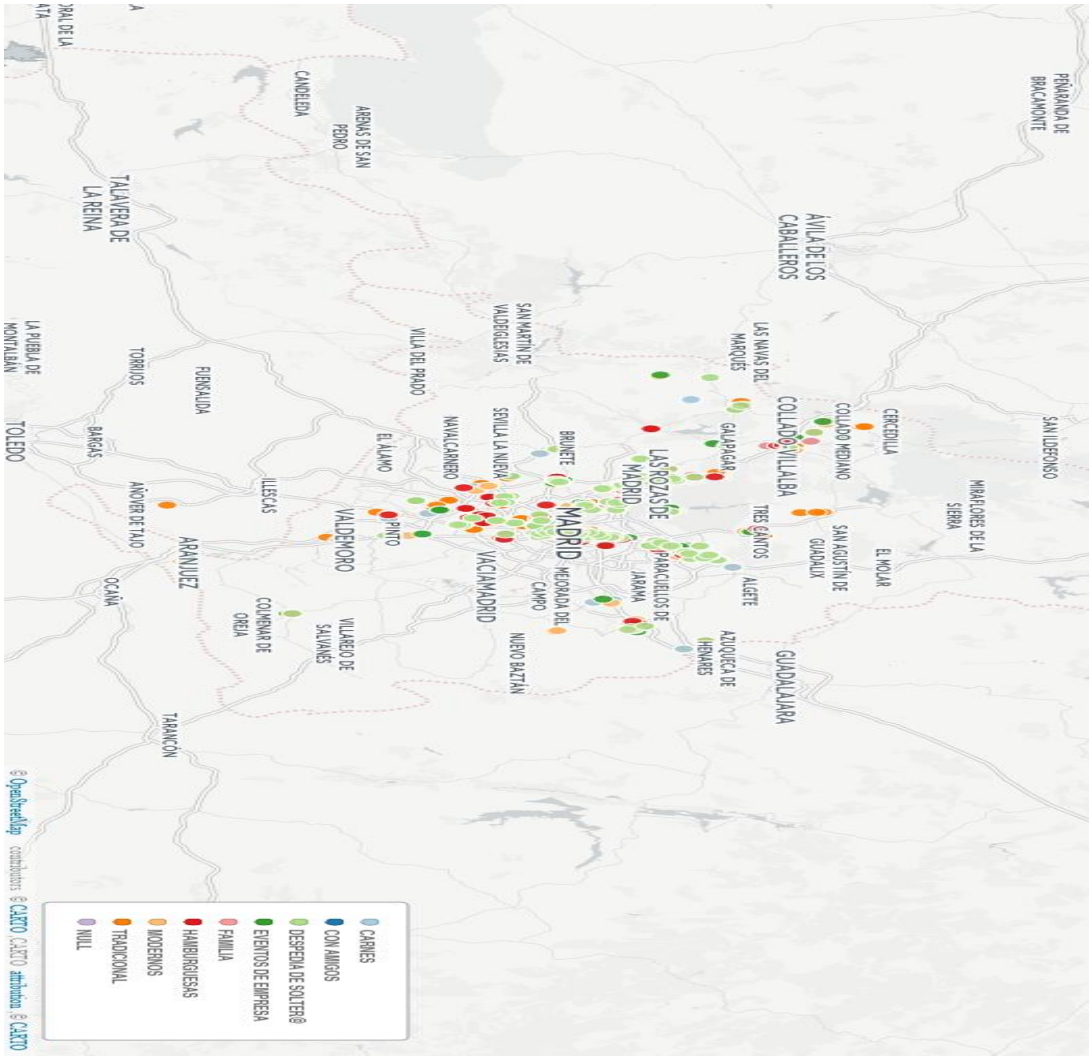


Ilustración 23: Distribución de restaurantes con precio medio alto

Tomando de nuevo Madrid como referencia al punto central podemos observar una clara diferencia entre los núcleos de restaurantes de precio alto en el área norte y sur. Contrastando la Ilustración 23: Distribución de restaurantes con precio medio alto con la Ilustración 21: Distribución de restaurantes con precio medio bajo podemos determinar claramente que la zona sur es más barata que la norte, por lo que se podría deducir un poder adquisitivo de la población menor.

5.1.6 Distribución de restaurantes por tag



La tendencia en la capital madrileña no difiere mucho de la catalana como se vio en el punto 5.1.6 Distribución de restaurantes por tag de la comunidad de Madrid. De nuevo predominan los tags sociales sobre el resto, aunque cuando más nos alejamos del centro, la diversidad de éstos comienza a hacerse notoria. Al sur podemos identificar un área donde los tags carnes y tradicional se ven con una frecuencia alta, de nuevo de manera similar al entorno catalán. También podemos apreciar en la zona norte que el tag social "familia "aparece como dominante, podemos considerar este dato como importante ya que aunque sea uno de los más frecuentes, en ningún área aparte de éste supera a otros tags de su categoría como "Despedida de soltero/a".

5.1.6 Distribución de restaurantes por nota media

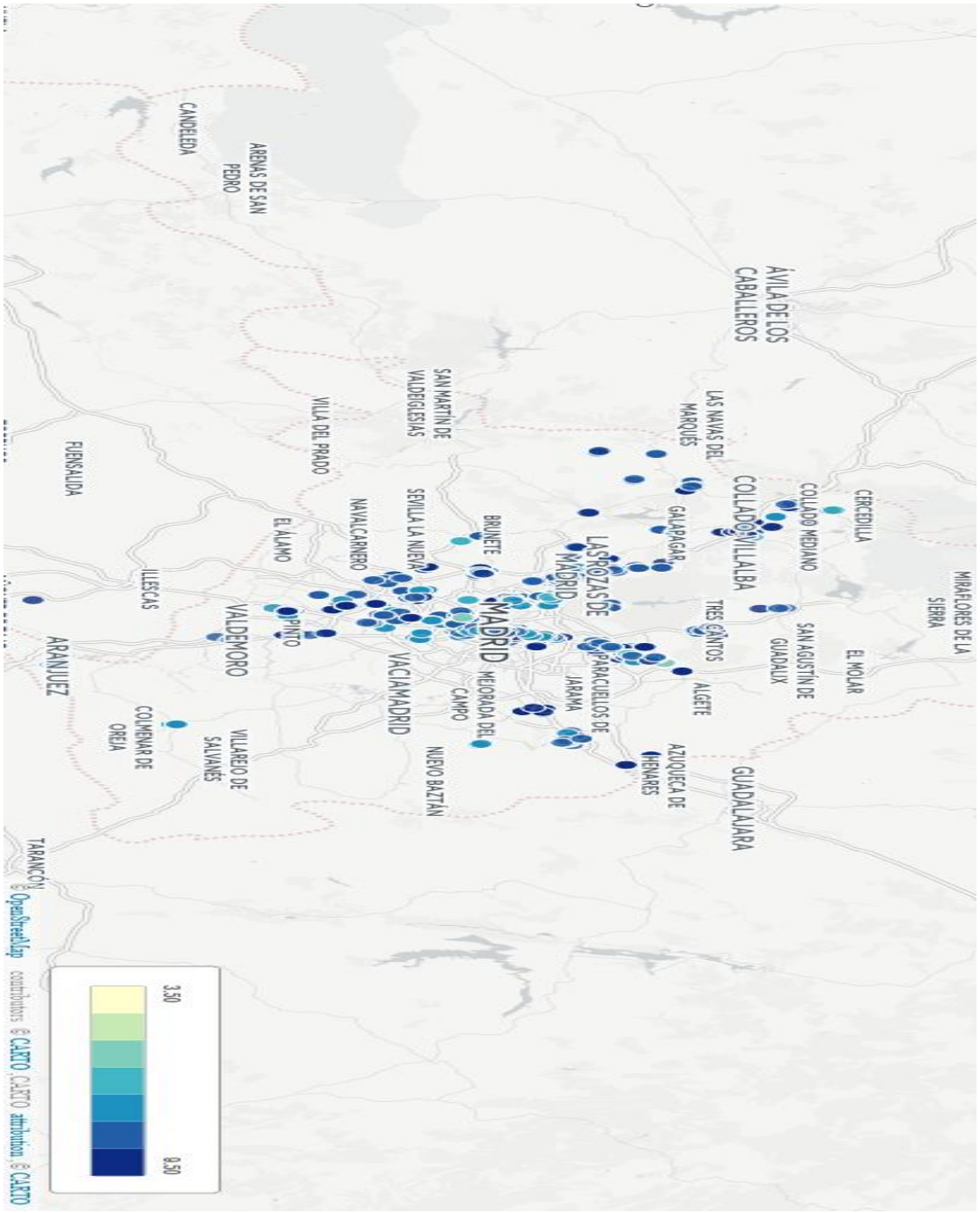


Ilustración 24: Distribución de restaurantes por nota media

A simple vista podemos ver en la Ilustración 24: Distribución de restaurantes por nota media que la conformidad del cliente es por lo general alta y la nota está distribuida de una manera más o menos uniforme, siendo la excepción el centro donde se

encuentra la aglomeración de notas más bajas. El rango de éstas a lo largo del mapa varía entre un 3.5 y un 9.5 de máxima.

Si comparamos la conclusión obtenida en este apartado con el punto de 5.1.2 Distribución de restaurantes en función del precio medio, podemos determinar que en relación calidad precio la zona sur es superior, ya que al ser la calidad similar el precio es la variable que determina esta conclusión y la peor es la zona centro, ya que hay mayor diversidad de precios, pero la calidad por norma general es bastante menor, rondando el 7.

5.2 Barcelona y alrededores

5.2.1 Distribución de restaurantes por densidad

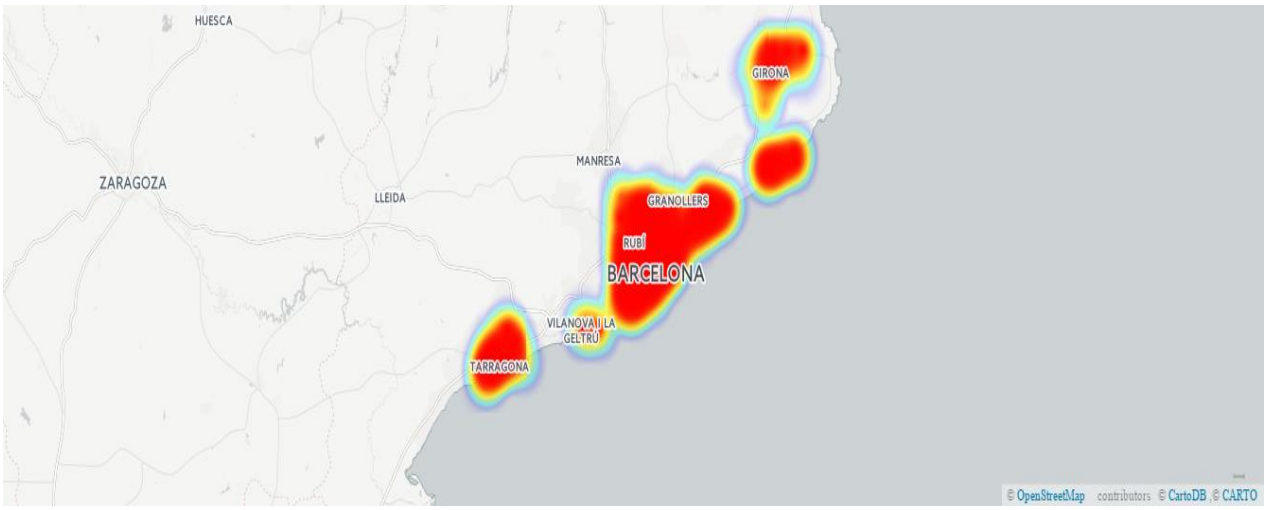


Ilustración 25: Distribución de restaurantes por densidad

Los datos obtenidos en la comunidad de Cataluña como se aprecia en la Ilustración 25: Distribución de restaurantes por densidad se encuentran distribuidos a lo largo de la costa, exceptuando las el área de interior que rodea a la capital y Girona.

5.2.2 Distribución de restaurantes por precio

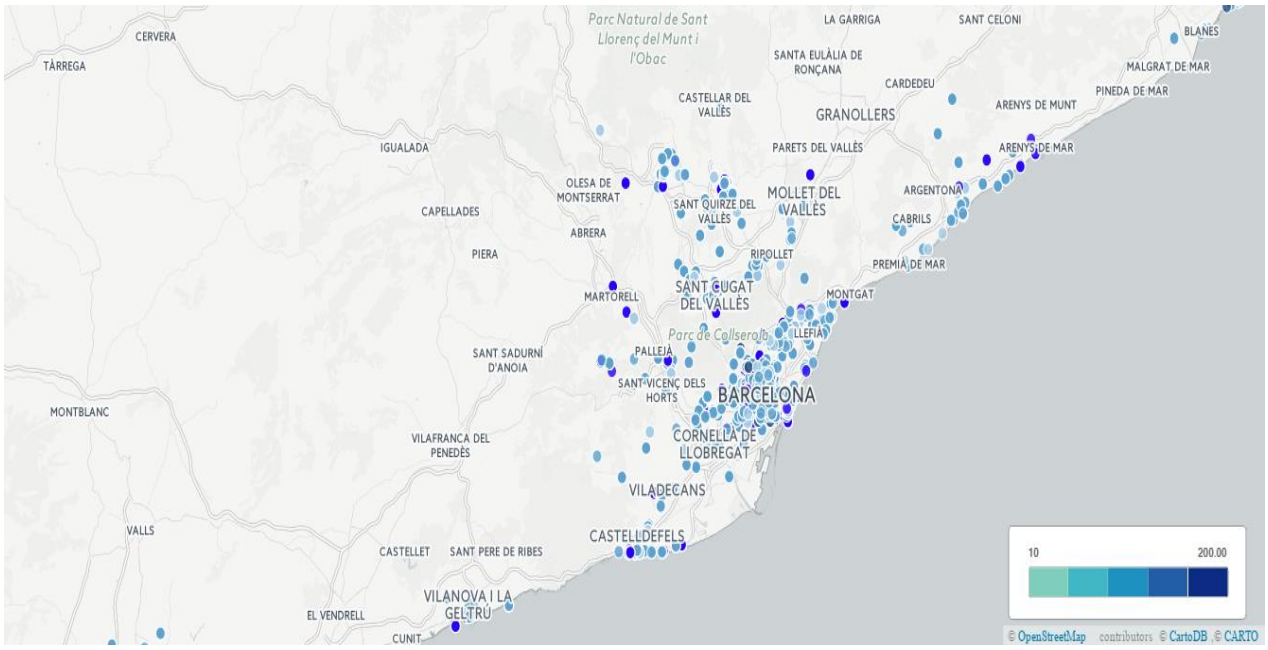


Ilustración 26: Distribución de restaurantes por precio

Podemos observar que la diferencia de precio medio entre el restaurante más caro y el más barato es bastante mayor que en la comunidad de Madrid, siendo el precio más bajo 10 euros y el más alto 200. En cuanto a la distribución, no podemos determinar que siga un modelo fijo, ya que los restaurantes baratos o de precio medio representan la mayor parte del dataset, estando los restaurantes de precio alto repartidos de manera equilibrada, entre los mayores núcleos de población.

5.2.3 Distribución de restaurantes por tag

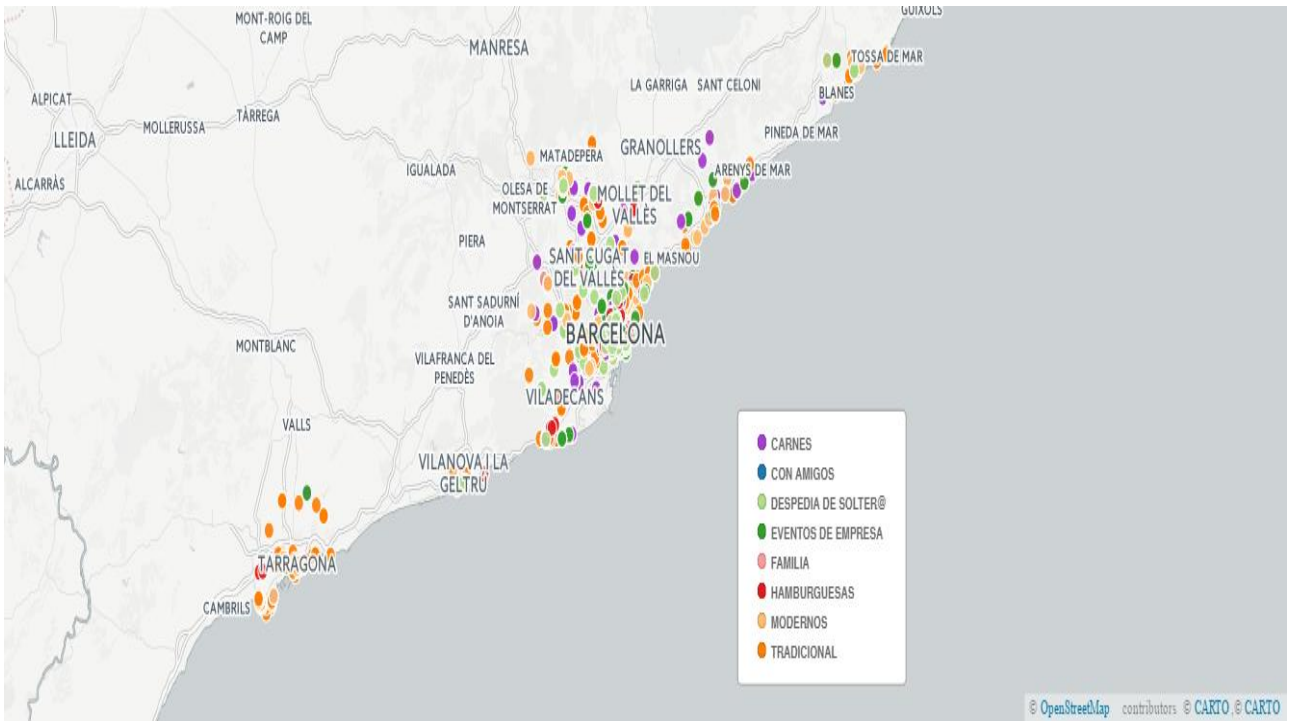


Ilustración 27: Distribución de restaurantes por tag

Con respecto a la distribución de los tags más populares representados en la Ilustración 27: Distribución de restaurantes por tag, podemos determinar que en la capital catalana predominan los restaurantes que buscan promocionarse con una estrategia social, es decir organizando eventos, sea personales o para empresas. Aunque hemos de tener en cuenta que, al ser la capital catalana, aquí se encuentra la mayor diversidad de restaurantes.

Subiendo un poco hacia la zona de Mollet del valle, la tendencia comienza a cambiar y vemos como protagonistas los restaurantes de carne, tradicionales e incluso modernos.

Al alejarnos de Barcelona, observamos que en la zona sur (Tarragona y alrededores), predominan los restaurantes tradicionales.

Lo que podemos determinar es que la capital es el punto con mayor densidad de población, por lo que existe una gran variedad y cantidad de restaurantes, y los propietarios a parte de la especialización del restaurante (moderno, carnes, chino, etc…) han de optar por una estrategia social, mientras que cuanto más nos alejamos de la capital el cambio de tendencia es claramente visible, los restaurantes buscan la tradicionalidad, la especialización en la comida típica y la buena carne.

5.2.4 Distribución de restaurantes en función de recomendaciones

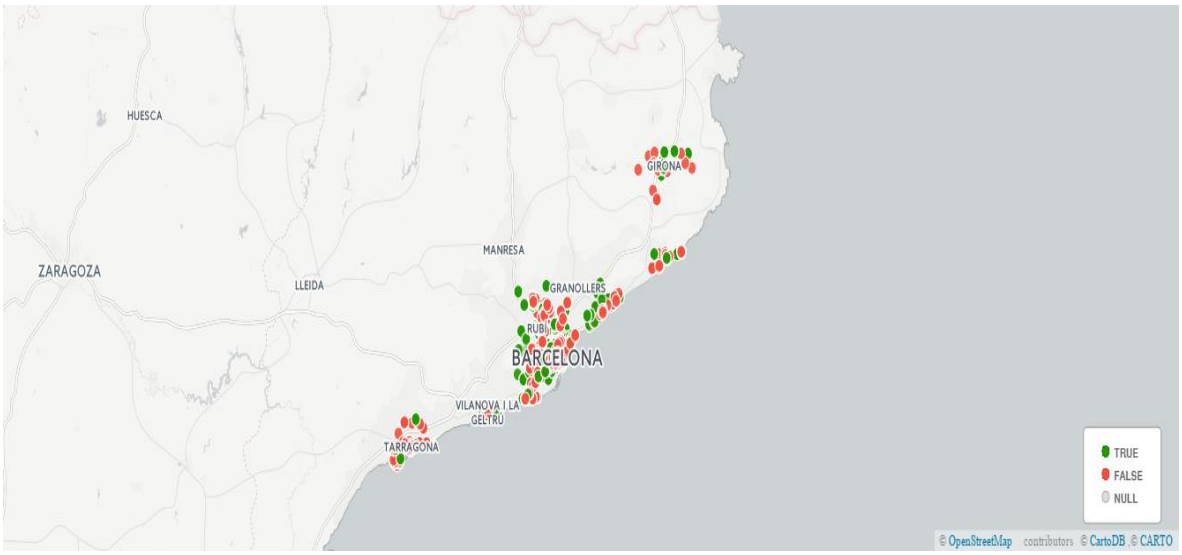


Ilustración 28: Distribución de restaurantes en función de recomendaciones

El gráfico nos muestra los restaurantes recomendados por los usuarios, siendo representados en color verde los recomendables y en rojo los que no lo son.

Podemos observar como en general, en la zona costera los clientes no recomiendan los restaurantes en un gran porcentaje, lo mismo ocurre en Tarragona y en el norte de Barcelona. Mientras que entre las zonas más recomendables para ir a comer se encuentra la zona sur de Barcelona y la zona interior que limita con la costa de Granollers. Por otra parte, vemos que hay zonas donde las críticas están igualadas como por ejemplo en Girona o la zona sur de ésta.

6 Comentarios

6.1 Visualización

Si aplicamos a los comentarios TF-IDF [18] , método por el cual intentaremos representar las palabras más influyentes en un documento, teniendo en cuenta toda la colección, obtenemos la siguiente representación de los resultados.

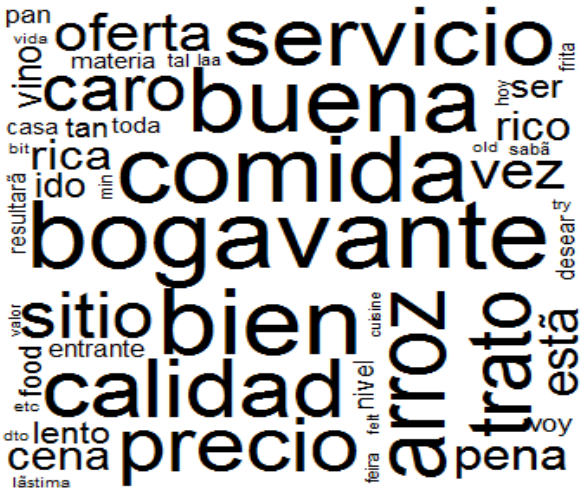


Ilustración 29: wordcloud de TF-IDF

Podemos observar que las entre las palabras más destacadas obviando las referentes a comidas son, por norma general, consideradas de carácter positivo como bien, bueno, rica… siendo la mayor crítica al precio con la aparición de caro, por lo que la conclusión es que la gente es poco crítica, y esto será un factor determinante a la hora de analizar lo relativo a los comentarios.

Decisión por tags: Aplicando TF-IDF a comentarios seleccionados por tags, podemos tener un criterio relevante para elegir un tipo de restaurante u otro.

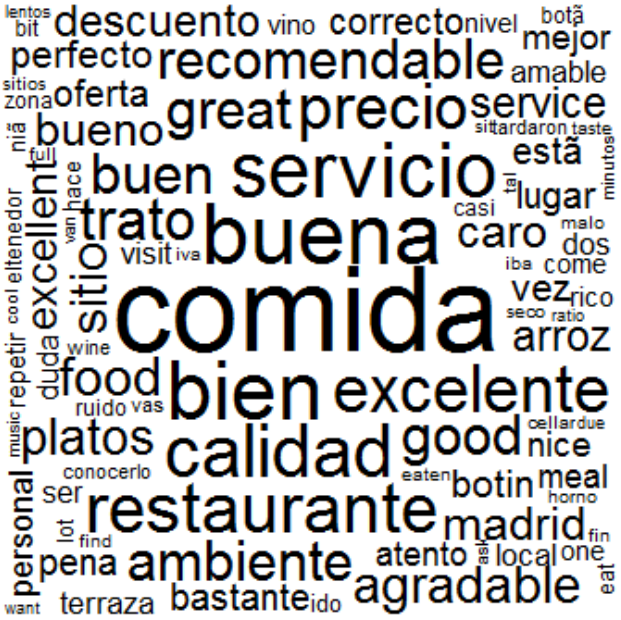


Ilustración 30 TF-IDF de "Moderno"

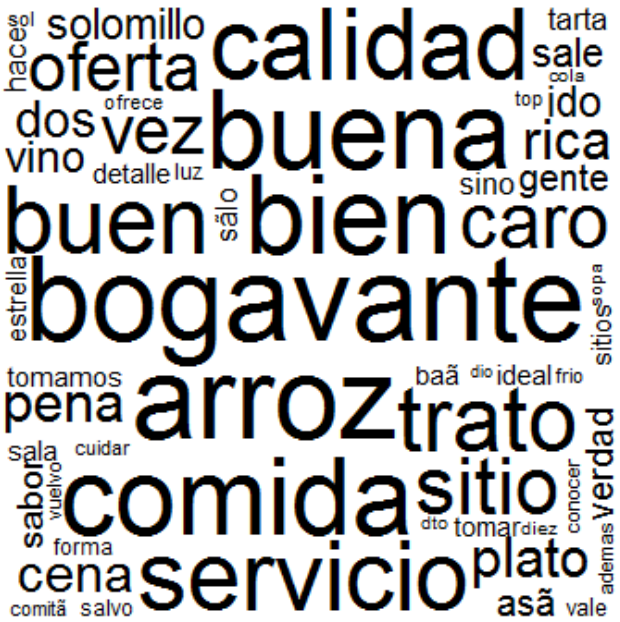


Ilustración 31 TF-IDF de "Tradicional"

A simple vista podemos observar que ambos restaurantes tienen una valoración positiva, como cabía esperar tras analizar el global de los comentarios, pero además entrando un poco más en detalle, podemos observar la cantidad de palabras de habla inglesa que son representadas en la Ilustración 32, por lo que podemos interpretar que los restaurantes modernos tienen una mayor acogida por parte del cliente extranjero. Además en las conversaciones de los restaurantes Tradicionales, se valora mucho más la comida, es decir los diferentes platos son repetidos en múltiples comentarios llegando a ser las palabas con más frecuencia, mientras que en los restaurantes modernos lo que aparece con mayor frecuencia son palabras relacionadas

con el ambiente y palabras genéricas con respecto a la comida ejemplo de esto son las palabras ambiente, amable, terraza, sitio, service, local, etc.. Con respecto a críticas comunes, como era de esperar vuelve a aparecer el precio, siendo la palabra “Caro” una de las más frecuentes en ambos datasets.

6.2 Modelos lineales

6.2.1 Estimación de precios en función de tags

Para realizar el modelo lineal del precio en función de los tags, hemos de reorganizar el dataset de tal modo que cada precio medio de restaurante tenga asignado todos los tags del restaurante, una vez hecho esto podemos pasar a realizar el modelo. Para poder interpretar los modelos se han de tener en cuenta los siguientes valores: Adjusted R-Squared o valor R cuadrado nos indica cómo de fiable es nuestra estimación con valores comprendidos entre 0 y 1, siendo 0 poco fiable y 1 muy fiable. Además, tenemos que tener en cuenta el valor de la tabla *Intercept,* pues este valor nos da el precio base de un restaurante en su dataset, este precio se verá incrementado o disminuido dependiendo del valor del *Estimate* asignado a cada tag. Y por último tenemos que tener en cuenta el valor P. Este indicador nos indica cómo de importante es una variable para la estimación, cuanto más próximo sea a cero, mayor será la relevancia de la variable.

6.2.1.1 Barcelona

En el caso de Barcelona el resultado de aplicar el modelo lineal es el siguiente.

```
Call:
lm(formula = lmtags, data = test.df)

Residuals:
    Min      1Q  Median      3Q     Max
-16.252  -2.868   0.000   2.230  28.268

Coefficients: (48 not defined because of singularities)
                                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                               21.20894    7.51836   2.821  0.00562 **
CataluÃ.a                                      NA         NA      NA       NA
Best.of.terrazas                           6.73502    3.43719   1.959  0.05242 .
Familia                                    0.37379    2.04512   0.183  0.85529
Tradicional                                2.35386    2.40184   0.980  0.32908
Diego.de.LeA-n                                 NA         NA      NA       NA
Plaza.de.Toros                                 NA         NA      NA       NA
Del.Norte                                      NA         NA      NA       NA
Belga                                          NA         NA      NA       NA
Camboyano                                 17.23373   16.26835   1.059  0.29161
AlemÃ.n                                   -1.26037    3.60752  -0.349  0.72743
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.92 on 118 degrees of freedom
Multiple R-squared:  0.8233,    Adjusted R-squared:  0.5838
F-statistic: 3.437 on 160 and 118 DF,  p-value: 6.306e-12
```

Ilustración 32 resumen lm Barcelona

Podemos observar que el modelo tiene un valor R-squared cercano al 0.6 sobre 1 de fiabilidad. El precio de partida de un restaurante son 21,20 euros de

acuerdo a la estimación subiendo o bajando el valor acorde con la siguiente tabla, por ejemplo una comida en el barrio gótico implica una subida de 49 euros sobre el precio, mientras que comer cerca de la plaza Cataluña reduce el precio en 19 euros.

	Estimate	Pr(>|t|)
GastronÃ³mico	9.82847232	0.001621974
Bodas	12.17902247	0.003073829
Casa.BatllÃ³	24.81194473	0.004933543
(Intercept)	21.20893547	0.005619212
Platos.vegetarianos	9.05278927	0.010568906
Paseo.de.Gracia	-9.48435102	0.014391807
Best.of.grandes.chefs	18.53865594	0.015550147
Pasta	7.21799996	0.034249209
Pulpo	8.02129111	0.041154129
Pinchos.y.Tapas	-4.60408177	0.049832019
Best.of.terrazas	6.73501857	0.052417511
Poblenou	26.31442418	0.065184490
Barrio.GÃ³tico	49.60023363	0.067107812
RomÃ.ntico	5.05137682	0.071039882
GÃ²tic	-11.39226103	0.072637443
Huevos.Rotos	-6.70066427	0.073407397
GrÃ.cia	4.77632497	0.077899936
Parque.GÃ.ell	-21.87905283	0.086546440
La.Ciutadella	-14.88423386	0.086889672
IranÃ.	29.56728546	0.088127547
Barceloneta	7.03751963	0.118486961
Best.of.vida.nocturna	5.92827878	0.119843302

	Estimate	Pr(>|t|)
Best.of.vida.nocturna	5.92827878	0.119843302
Bautizos	-8.03509958	0.126379341
Kosher	43.59428052	0.127833327
Gran.VÃ.a	18.99328241	0.131248338
Catedral	-38.00546302	0.149446424
Comuniones	6.54527983	0.155637413
Tartar	-5.15318124	0.166064017
Vallcarca	11.56375844	0.167734906
Cerca.del.mar	-4.23622763	0.172271447
Certificado.de.Excelencia.TripAdvisor.2015	2.85237414	0.178837613
Plaza.CataluÃ.a	-17.45761681	0.192358908
Despedida.de.solter.	-3.78312241	0.193557136
La.Sagrada.Familia	-23.21383078	0.196050598
Soles.Repsol.2016	19.97365101	0.197544779
Port.Vell	-7.89481638	0.197582102
Parque.de.la.Ciudadela	-8.67273935	0.214183256
De.negocios	2.72500107	0.217609522
Ciutat.Vella	7.06928216	0.242268982
PortuguÃ.s	13.15604995	0.245283635
Vegetariano	-13.73741796	0.248967867
Paralelo	-7.73583569	0.250750343
EstaciÃ³n.de.Sants	6.11914511	0.250989504

Ilustración 33: Tabla de estimates/p values Barcelona

6.2.1.2 Madrid

En el caso de Madrid el resultado de aplicar el modelo lineal es el siguiente.

```
San.Bernardo                                    NA         NA      NA       NA
MarroquÃ.                                  4.49964    7.26030   0.620  0.53980
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.718 on 32 degrees of freedom
Multiple R-squared:  0.9128,    Adjusted R-squared:  0.3647
F-statistic: 1.665 on 201 and 32 DF,  p-value: 0.04358
```

Ilustración 34: resumen lm Madrid

Podemos observar que el modelo presenta una fiabilidad baja, de acuerdo con el valor R-cuadrado, además el precio base para un restaurante en Madrid son 36,7 euros, precio sorprendentemente superior al de Barcelona, teniendo en cuenta que los restaurantes más caros se encuentran en la capital catalana. En la columna Estimate de la Imagen 34 podemos observar qué tags dan más valor teniendo en cuenta que estos tags son relevantes para la estimación ya que el valor P es bajo. Así por ejemplo el tag Best of grandes chefs incrementa el precio medio en 56 euros y tags como brasileños lo reducen en 56.

	Estimate	Pr(>|t|)
Best.of.grandes.chefs	56.343147	0.001073044
Hotel	22.212533	0.013373712
Barrio.Salamanca	-40.210067	0.018460989
Brunch	-99.739456	0.041019783
Bacalao	15.184039	0.054055830
Cibeles	-29.802939	0.056183648
RomÃ.ntico	18.499740	0.056476370
BrasileÃ.o	-64.515848	0.059720710
El.PlantÃ.o	-21.681406	0.063937931
Paseo.del.Prado	121.644699	0.073023401
Santiago.BernabÃ.u	-37.979200	0.074846327
Couscous	-26.459799	0.085141720
Best.of.come.por..15.â..	-13.189953	0.085809460
Castellana	31.683848	0.096347765
	Estimate	Pr(>|t|)
Mercado.de.San.Miguel	40.505294	0.100090290
Paradores	-55.874517	0.103240849
Plaza.de.La.Villa	-121.203516	0.128971969
Mercado.gastronÃ³mico	125.816452	0.131727305
Atocha	25.212308	0.132203674
De.Autor	11.033990	0.134253547
Centro.comercial	45.116532	0.139033483
Cuzco	-19.067654	0.142928037
Alonso.MartÃ.nez	-11.315021	0.145492719
GastronÃ³mico	8.961704	0.145822066
JerÃ³nimos	-226.158688	0.148619637
TetuÃ.n	-36.738465	0.150187867
Museo.Thyssen	-73.459972	0.168551064
(Intercept)	36.673506	0.170825344

Ilustración 35: Valores estimados para lm de Madrid

7 Conclusions and Future Work (Eng)

In summary, the conclusion that can be extracted about Data Analytics and Big Data technologies is that a dataset offer almost unlimited opportunities for new projects, researches and applications.

Furthermore, Big Data techniques are not limited to big datasets, the meaning of Big Data technologies is that using them, you can actually provide scalability, giving the opportunity to test in small datasets and then swap to a large dataset without a huge effort in new analytic algorithms and avoiding several problems with the storage system.

Another fact that a Big Data developer have to take into account, is that with the popularity that actually is gaining, it becomes much easier to learn about it having a huge span of possibilities and new ones created, this is in part thanks to the good reception given to this technology in modern enterprises, with proven economical results.

Essentially, the dataset retrieved from “El Tenedor” has a huge potential, and the analytics and applications obtained can be correlated with other data or done in a different way. Moreover we have a lot of unused data that has not been processed due to the lack of time.

In a nutshell, the scope of the project has been sucesfully fulfilled, in compliance with the following objectives:

Retrieve Data from a website, dealing with the problems that represents a real server such as massive petition rejection or variable HTML code. The technologies used to fulfil this requirement were, a Python algorithm composed by two modules, one for the HTML code download and the other one to parse the content and extract the relevant data.

We have been able to set a workspace where the crawler algorithm, the storage system and the analytics program are successfully connected and working properly. The first step is connect the crawler algorithm with the database system, we have chosen mongoDB since it uses JSON format, which is compatible with the parsing module.

The experiments shows social and commercial tendencies in the Madrid and Barcelona restaurants, which is the basis of “el Tenedor” social network, helping it to grow and improve, and at the same time helping the users which is the main objective of the analysis. Some of this analytics include map representation of relevant data, such as restaurant locations by tag or by score. We have also used algorithems as TF-IDF over the commentaries which shows the tendencies of the clients. Or linear

estimators in order to prove the relation between the price and the restaurant tags.

Certainly, we can give some applications to the analytics presented in this project, with or without external data such as a new restaurant locator, the offer restaurant route with gourmet areas and not recommendable ones from a gastronomic point of view, an application that shows the level of trust on a comment…Some detailed examples could include the following.

- New restaurant locator: Adding a new entries to our dataset such as an economic research of the per capita rent divided by zones of the analyzed cities and also with a real state online agency, to have the prices and locations of new emptly locals for sale. With all this information we can develop a market study for those who wants to open a new restaurant.

- Your favourite food app: We have the cards of every restaurante in the dataset, and it is actually unused. Combining this information with the commentaries especially about the food, prices and ratings, we can recommend restaurants as function of a plate.

- User evaluator: With the information we have available, such as the maestry of the user commenting which shows how participative the user is in the page, also the name of the user, and we can also extract the inormation about how many times replied to other users and how possitive or neggative tends to be on comments. Allowing us to develop some user classification with those parameters.

Bibliografía

[1] «IBM,» [En línea]. Available: https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html.

[2] BOE, «LOPD,» 12 Septiembre 2016. [En línea]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-1999-23750.

[3] «http://www.json.org/json-es.html,» [En línea]. Available: http://www.json.org/json-es.html.

[4] «http://schoolofdata.org/handbook/recipes/scraper-extension-for-chrome/,» [En línea]. Available: http://schoolofdata.org/handbook/recipes/scraper-extension-for-chrome/.

[5] «https://www.screamingfrog.co.uk/seo-spider/,» [En línea]. Available: https://www.screamingfrog.co.uk/seo-spider/.

[6] «K-means,» 20 Agosto 2016. [En línea]. Available: https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm.

[7] «word cloud,» 20 Agosto 2016. [En línea]. Available: https://www.r-bloggers.com/building-wordclouds-in-r/.

[8] «gplot,» 20 Agosto 2016. [En línea]. Available: https://cran.r-project.org/web/packages/gplots/index.html.

[9] «Baricentro o Centroide,» 15 Septiembre 2016. [En línea]. Available: http://mitecnologico.com/igestion/Main/CalculoDeCentroides.

[10] http://www.eltenedor.es/. [En línea]. Available: http://www.eltenedor.es/.

[11] «w3api,» [En línea]. Available: http://www.w3api.com/wiki/Java:HttpServlet.doGet().

[12] «HTTP,» [En línea]. Available: https://www.w3.org/Protocols/rfc2616/rfc2616.html.

[13] «https://www.crummy.com/software/BeautifulSoup/bs4/doc/,» [En línea]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/. [Último acceso: 15 Julio 2016].

[14] «https://pypi.python.org/pypi/pycurl,» [En línea]. Available: https://pypi.python.org/pypi/pycurl. [Último acceso: 16 Julio 2016].

[15] «https://docs.python.org/2/library/re.html,» [En línea]. Available: https://docs.python.org/2/library/re.html. [Último acceso: 16 Julio 2016].

[16] «https://api.mongodb.com/python/current/tutorial.html,» 16 Julio 2016. [En línea]. Available: https://api.mongodb.com/python/current/tutorial.html.

[17] «https://stat.ethz.ch/R-manual/R-devel/library/base/html/lapply.html,» [En línea]. Available: https://stat.ethz.ch/R-manual/R-devel/library/base/html/lapply.html.

[18] «http://www.tfidf.com/,» [En línea]. Available: http://www.tfidf.com/.

ANEXO I Summary

Introduction

The aim of this project is to analyze, study and represent the data obtained from a social network, “El Tenedor”. Taking as steps the extraction of the data, storage and filtering of data, analyze the useful data and represent the results.

Crawling

In order to extract data from the website of "El Tenedor" two modules will be required, the first is a web extraction program to parse content among other useful information, both of them programmed in Python.

Extraction module html content: Web pages mainly use a GET function in order to send HTTP requests and forms to a server, which implies that part of this information is shown clearly in the URL, so in most cases we can guess which is going to be the next page to Crawl so we can simulate a request in order to download the HTML content.

For data efficiently, we need an algorithm that allows us to get a lot of html content mechanized way, this is known as a web spider or web crawler. The basis of the procedure is to access the links from the website iteratively inspected and organized manner.

Access to the links is determined by the design of the website, so we can take advantage of css classes, or the elements HTML to locate the points of each page which can get some benefit, for that purpose we use our library BeautifulSoup, this library allows us to parse the downloaded document (lxml parser), or find matches with regular expressions. Thus redirects to find all pages that interest you to download the code. Obtaining data with massive queries to a server, certain problems may happen, such as massive rejection of requests with the same origin or and not being a web browser arise. We will use pycurl library functions, they allow us to simulate a web browser, in our case will be Mozilla Firefox, and we will do random active wait between requests to different URLs establishing a user-agent header to meet the standard of the petition.

Focusing on filtering an html document, we will have all the documents downloaded in folders properly organized, as previously mentioned we will use an lxml parser since is the one that fits better with the OS used, Windows 7. This parser will be done with BeautifulSoup, a python library for this purpose.

Once obtained the Soup object, which is basically the parsed document itself, is easy to extract the content of a specific CSS class, or a list with all of the content contained in the whole document limited by this classes. Is also viable to find for an id,

regular expression or any html element, since the Beautifulsoup library allow us to find them with a simple call to their functions.

Once parsed almost the whole document, we can find that a huge part of the information located in the header of the document, delimited by the tag <head> as a part of the JavaScript code, in a JSON format, or mixed with text that does not give us valid information.

For this purpose, “re” Python library offers a somewhat tedious solution to the programmer, since for each variable, or set of them has to be designed an expression.

Once the regular expression is designed, the “compile” function will save in case of correct syntax of that expression, which later will be used by the findAll function, which as in the case of BeautifulSoup, we will return a list of all matches in the HTML document.

Already obtained all the variables needed, they have to save in the database so they can be easily accessible, therefore we take advantage of Python has dictionaries whose syntax is the same as a file format JSON, which it is easily importable by MongoDB.

Once we have obtained the JSON dictionary, we have to insert the restaurant information into the MongoDB database, to do so we need a server-client connection with the DB, python makes it simple just by calling a connector function. Once connected a single call to a function is required to add a new entry or collection to our dataset.

Dataset

The dataset is divided into two groups containing the information of the restaurants in Madrid and Barcelona.

The amount of data obtained for the analysis is 1.96 GB for Madrid, with 1903 restaurants and 1.29 GB for 2377 Catalonia restaurants.

After a thorough parsing of the information obtained, we obtain useful data for the study, in JSON format, the amount of these data reaches 92.3 MB and 63.6 MB for Madrid and Catalonia respectively. Each restaurant has variables structured as follows

The main variables contain information about two differentiate topics, the restaurant itself; including the location, five kind of rates, price, card with price, type of restaurant and tags and many more entries. The other kind of data is related to the user containing his opinion, score given to the restaurant and expertise in the page.

The dataset must be imported later on by the statistic tool. R allow us to do it with a connector called *Rmongo.* One of the complexities of working with data is ordering datasets, to do so we will use functions such as lapply to order the data directly

extracted. Usually the objective is to obtain a DataFrame which is a list of vectors with equal length and many functions in R requires it to work properly.

As example of how the data was received in R to be processed and analyzed we have the following dataset:

```
> db.barcelona.findOne()
{
        "_id" : 1,
        "comment" : [
                "Nosotros solo comimos sushi y no nos gusto, nada elaborado y co
n mucho arroz.",
                "He ido a este restaurante varias veces y ésta era la primera co
n promoción. Puedo decir que tanto el servicio como la comida es de la misma cal
idad. Excelente. Muy recomendable para cualquier tipo de cena.",
                "reservé para ir al del puerto pero me avisaron con antelación q
ue no podía ser posible allí porque estava cerrado, que podía ir al que tienen e
n Francesc Layret manteniendo la promo , fuimos y como siempre estuvo todo genia
l , (para mi mejor sushi calidad precio de Badalona , eso sí ,con la promo) .",
                "Tienen una carta variada y de calidad. Pero el precio es un poc
o caro, aunque suelen tener casi siempre un descuento por promoción "
        ],
        "rating" : null,
        "restaurantId" : 43278,
        "categoryName" : "Japonés",
        "saleTypeIsMenu" : 0,
        "maestry" : [
                "Rey Gourmet",
                "Explorador/a",
                "Conquistador/a",
                "Conquistador/a"
        ],
        "saleTypeId" : 1073580,
        "TotalRate" : "5,5",
        "cityZipcode" : "08912",
        "idCityZipcode" : 95959,
        "saleTypeMaxPartySize" : 0,
        "message" : "Reserva sin promoción",
        "thumbnailUrl" : "http://u.tfstatic.com/restaurant_photos/278/43278/43/8
0/aroma-kaori-port-badalona-vista-cocina-bdd19.jpg",
        "id" : "43278_2",
        "saleTypeDiscountAmount" : null,
        "name" : "Aroma Kaori Port Badalona",
        "card" : [
                "Entrante",
                "5,8 ?",
                "Patatas bravas",
                "9,95 ?",
                "Yakisoba",
                "9,8 ?",
                "Calamares andaluza",
                "Plato",
                "10,95 ?",
                "Tempura vegetal",
                "18,45 ?",
                "Sushi moriawase",
                "9,8 ?",
                "Filete de ternera con foie",
                "Postre",
                "5,8 ?",
                "Cuatro texturas chocolate",
```

Ilustración 36: MongoDB data in Json

Data analysis

Most of the technologies involved in data mining algorithms are used to rank, rate or dataset separate words. These algorithms are well known, so they are sufficiently developed to provide higher performance than a proprietary algorithm.

This is the point where R is critical to the development of this project, the basic data mining capabilities are supported by R. Addition R has libraries that greatly enhance these capabilities. As is the case of *stats*, *plyr* or *tm*; which they are libraries that are to be added to our environment.

The positioning of the words is a point that must be covered, for this purpouse algorithms include mathematical models that attempt to rate and to discriminate

twords, to thereby obtain the most relevant, depending on various factors, taking into account both the document itself as all the collection.

In order to realize this positioning, algorithms take as an indispensable parameter frequency term in the document and all of them.

The most relevant algorithm for this project is Term Frequency Inverse Document Frequency known as TF-IDF.

This algorithm is mainly used to extract keywords within a large collection of texts. The TF-IDF algorithm computes the TF-IDF rate for each term and document. The TF-IDF value increases proportionally to the number of times a word appears in the document, but is compensated with the word frequency in the collection, in order to avoid confusion with common words that does not apport useful information to the research.

Starting with analysis, we will proceed to apply simple studies to the data, for example, taking just one parameter, tags, just knowing the frequency we can guess the tendencies in each City, and clearly see that Barcelona is quite a lot focused on tourism meanwhile Madrid have a huge enterprise potential.

Now analyzing the all the rates and comparing them, we can know what aspects in a restaurant people criticize more, and which ones are less influential in the total rate, we discovered that people tended to be more optimistic with the service, that may be because the personal factor use to be great in our Capitals, and we can say the same about food. But when the factor money is taken into account, people is less generous rating, that means that people thinks that eating is a little bit overpriced, not too much since the score even being quite under the mean, is still positive.

Studying and comparing quality and price, just by a correlation of variables and a plot, we can observe that the quality is pretty good for the Spanish restaurants and the price is equilibrated, this means that the quality does not depend on the price, moreover we can affirm that there are no bad restaurants with a huge price, and the best restaurant, is also the most expensive in the whole dataset.

Furthermore, we have extracted enough information to create an estimator based on the tags of a restaurant, that have proven to be more efficient in Barcelona than in Madrid. To do so we have used a linear model, which tries to estimate a variable as a function of many others. To properly use it we have to divide the dataset in two, the first one is the train set, which is used to obtain the linear model, and a test set to prove the efficiency of the model developed.

Moreover, we have represented some variables our datasets geo localized, which is really useful in terms of analytics. We have used CartoDB to fulfil this objective, which is a web application that allow the user to upload a dataset, modify it and represent it. For example, we have represented Madrid and Barcelona restaurants by price, discovering which places are the

cheapest and in opposition, the most expensive ones. In addition, we have compared the located tags, and we have seen that social tags in capitals tend to be concentrated in almost every area near the center of the City, and the farther you go from that point, food tag concentrations start appearing. By last, we had the information of the user recommendations, which is very useful since we can determine whether an area is recommendable or not, as we have shown in the Barcelona case, where for example, coast places where not very recommendable to go.

ANEXO II Planificación y recursos

A.II: Planificación

La duración estimada del proyecto son 9 meses, comenzando en diciembre de 2015 y finalizando en septiembre de 2016. Para el correcto desarrollo del proyecto hemos de administrar el tiempo de manera eficiente en paquetes de trabajo (P) compuestos por tareas.

P1. Estudio e investigación de las Tecnologías para Big Data y posible dataset:
- T1.1 Estudio de los conceptos y tecnologías para Big Data.
- T1.2 Búsqueda de un dataset válido para el estudio.
- T1.3 Preparación de la estrategia de crawling.

P2. Crawling de datos:
- T2.1 Diseño del prototipo de un programa Crawler.
- T2.2 Desarrollo del módulo de descarga.
- T2.3 Desarrollo del módulo de parseo.
- T2.4 Test del crawler desarrollado
- T2.5 Preparación e instalación de sistemas de almacenamiento.
- T2.6 Crawl de datos.

P3. Configuración e instalación del entorno para el procesamiento de datos:
- T3.1 Selección y estudio del entorno.
- T3.2 Instalación y testeo con pequeños datasets.
- T3.3 Testeo con el dataset del proyecto.

P4. Procesado de datos:
- T4.1 Estudio de los datos dentro del entorno.
- T4.2 Comprobación de resultados.

P5. Representación de resultados:
- T5.1 Estudio de medios de representación.
- T5.2 Representación de datos.

P6. Documentación y memoria:
- T6.1 Documentación parcial durante el desarrollo.
- T6.2 Estructuración e hitos.
- T6.3 Redacción de memoria.
- T6.4 Revisión y redacción.

P7. Preparación de la presentación:
- T 7.1 Elaboración de transparencias.
- T 7.2 Preparación de la presentación.

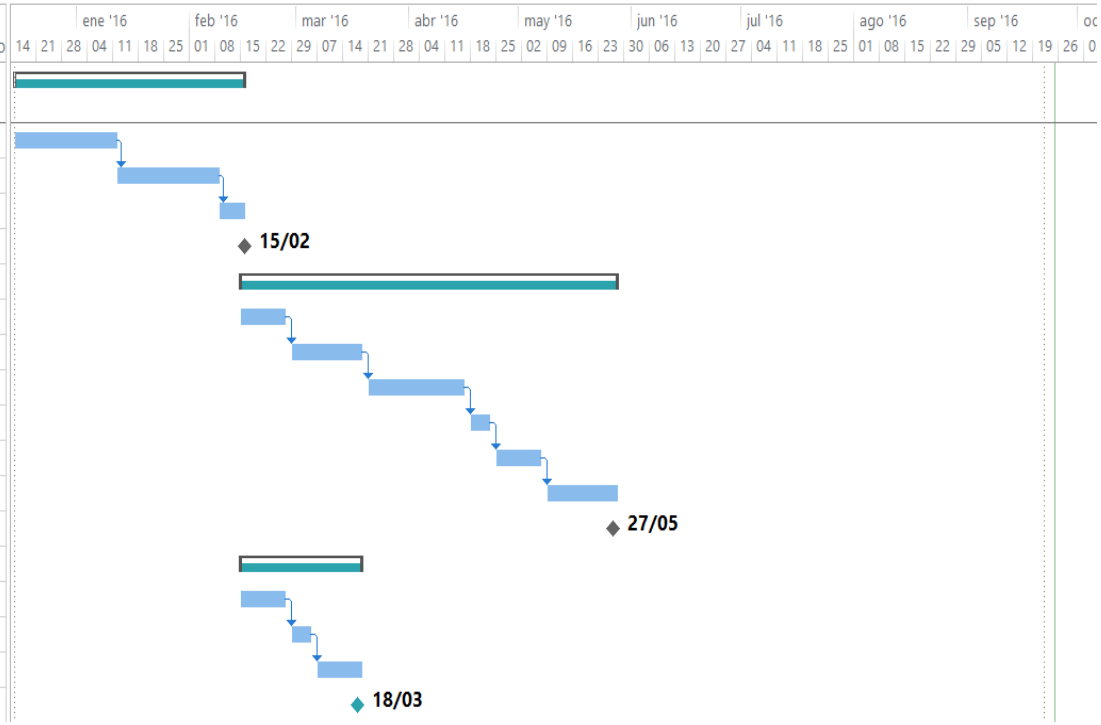
ESCALA D

enero | febrero | marzo | abril | mayo | junio | julio | agosto | septiembre

Comienzo mar 15/12/15 — Agregar tareas con fechas a la línea de tiempo — Fin mié 21/09/16

DIAGRAMA DE GANTT

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predeceso
1		**P1 Estudio e investigación**	**45 días**	**mar 15/12/15**	**lun 15/02/16**	
2		T1.1 Estudio de conceptos	20 días	mar 15/12/15	lun 11/01/16	
3		T1.2 Busqueda de dataset	20 días	mar 12/01/16	lun 08/02/16	2
4		T1.3 Estrategia de crawling	5 días	mar 09/02/16	lun 15/02/16	3
5		H1 Orientación del proyecto	0 días	lun 15/02/16	lun 15/02/16	
6		**P2 Crawling de datos**	**75 días**	**lun 15/02/16**	**vie 27/05/16**	
7		T2.1 Estudio de conceptos	10 días	lun 15/02/16	vie 26/02/16	
8		T2.2 Desarrollo de descarga	15 días	lun 29/02/16	vie 18/03/16	7
9		T2.3 Desarrollo de parseo	20 días	lun 21/03/16	vie 15/04/16	8
10		T2.4 Test del crawler	5 días	lun 18/04/16	vie 22/04/16	9
11		T2.5 Preparación de la BBDD	10 días	lun 25/04/16	vie 06/05/16	10
12		T2.6 Crawl de datos	15 días	lun 09/05/16	vie 27/05/16	11
13		H2 Dataset completado	0 días	vie 27/05/16	vie 27/05/16	
14		**P3 Configuración del entorno**	**25 días**	**lun 15/02/16**	**vie 18/03/16**	
15		T3.1 Selección y estudio	10 días	lun 15/02/16	vie 26/02/16	
16		T3.2 Instalación y test	5 días	lun 29/02/16	vie 04/03/16	15
17		T3.3 Test con dataset real	10 días	lun 07/03/16	vie 18/03/16	16
18		H3 Entorno funcional	0 días	vie 18/03/16	vie 18/03/16	



ESCALA D

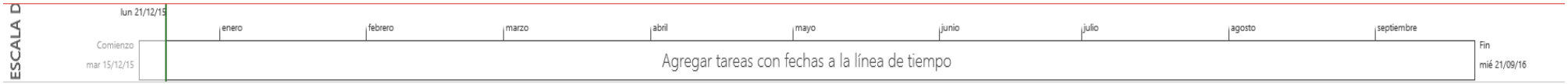
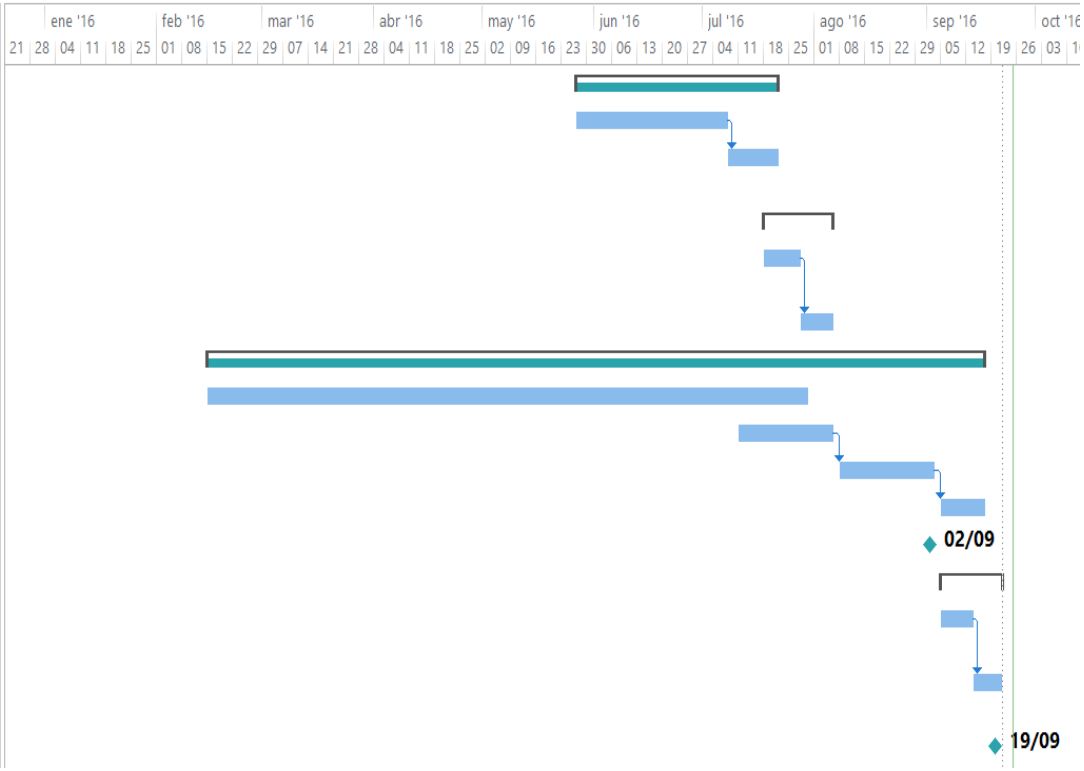


DIAGRAMA DE GANTT

		Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predeceso
19			**P4 Procesado de datos**	**40 días**	**vie 27/05/16**	**jue 21/07/16**	
20			T4.1 Estudio de datos	30 días	vie 27/05/16	jue 07/07/16	
21			T4.2 Comprobación de resulstados	10 días	vie 08/07/16	jue 21/07/16	20
22			**P5 Representación de resultados**	**15 días**	**lun 18/07/16**	**vie 05/08/16**	
23			T5.1 Estudio de representaciones	8 días	lun 18/07/16	mié 27/07/16	
24			T5.2 Representación	7 días	jue 28/07/16	vie 05/08/16	23
25			**P6 Documentación y memoria**	**155 días**	**lun 15/02/16**	**vie 16/09/16**	
26			T6.1 Docuentación parcial	120 días	lun 15/02/16	vie 29/07/16	
27			T6.2 Estructuración e hitos	20 días	lun 11/07/16	vie 05/08/16	
28			T6.3 Redacción de memoria	20 días	lun 08/08/16	vie 02/09/16	27
29			T6.4 Revisión y redacción	10 días	lun 05/09/16	vie 16/09/16	28
30			H4 Memoria completada	0 días	vie 02/09/16	vie 02/09/16	
31			**P7 Preparación de presentación**	**13 días**	**lun 05/09/16**	**mié 21/09/16**	
32			T7.1 Elaboración de transparencias	7 días	lun 05/09/16	mar 13/09/16	
33			T7.2 Preparación de presentación	6 días	mié 14/09/16	mié 21/09/16	32
34			H5 Presentación preparada	0 días	lun 19/09/16	lun 19/09/16	



A.II: Presupuesto

Un proyecto de investigación como el que se está describiendo requiere un presupuesto muy detallado y exhaustivo a fin de tener la máxima eficiencia posible de los recursos disponibles. Debido a esto, el presupuesto en el proyecto debe contener todos los costes, incluidos los posibles gastos inesperados, con el fin de lograr la mejor optimización posible.

El principal coste en el proyecto son los recursos humanos, ya que los materiales de trabajo son los ordenadores y el alquiler de un servidor.

En este proyecto, hay tres componentes del equipo: El director de la tesis, el codirector y el alumno.

El estudiante ha contabiliza las horas de trabajo como 4 horas por día durante 5 días por semana ordinarios durante 9 meses, que es un total de 720 horas en total.

Los tutores han dedicado una menor cantidad de tiempo, aunque se presupone una cualificación mucho mayor que la del alumno (lo que conlleva un coste por hora superior) rondando media hora semanal, estimando 90 horas en total.

Para estimar los salarios tendremos en cuenta la medida Persona-Mes(PM). Esta medida de tiempo indica la cantidad de tiempo que dedica un empleado a su trabajo. Normalmente son 160 horas mensuales, aunque en el caso de un profesor están fijadas 130 horas para investigación, ya que durante el resto han de dedicarlas a docencia. Teniendo esto en cuenta consideraremos 24000€/año para el alumno, 30000€/año para el director, siendo profesor adjunto y 40000€/año para el codirector, en calidad de profesor.

Para calcular el salario por PM se divide el salario entre los meses, con lo que resulta 2000€ mensuales para el alumno, 3000€ para el director y 4000 para el codirector. con lo que tendremos en cuenta los PM para cada trabajador, en el caso del alumno, serían 720 horas / 160 horas/PM resultando un total de 4,5PMs que al cambio en euros son 9000€. El director y codirector obtendrían un total de 0.7 PMs (90 horas/ 130 horas/PM). Siendo un total de 2100€ y 2800€ respectivamente.

Otros gastos pertinentes son todos los equipos utilizados durante el proyecto. Para todo el hardware utilizado durante el desarrollo de toda una vida de 3 años ha sido considerado y, en consecuencia, al premio del dicho equipo se ajusta proporcionalmente a la duración del proyecto. También se ha de tener en cuenta el alquiler del servidor como parte del hardware. Por último, las licencias de software no deben considerarse, ya que han sido utilizadas herramientas Open Source para la totalidad del desarrollo.

ID	Nombre	Descripción	Coste
S1	Salario del director	Costes del director	2100
S2	Salario del codirector	Costes del codirector	2800
S3	Salario del alumno	Costes del alumno	9000
S4	Ordenador del profesor	Coste de un tercio del valor real, teniendo en cuenta una esperanza de tres años.	350
S5	Ordenador del alumno	Coste de un tercio del valor real, teniendo en cuenta una esperanza de tres años.	300
S6	Alquiler del servidor	Coste de un veinteavo del valor real, teniendo en cuenta la esperanza de vida de 5 años y 3 meses de uso	300
	Coste total		14850

Ilustración 37: Tabla de costes